PROYECTO INTEGRADOR DE LA CARRERA DE INGENIERÍA NUCLEAR

ANÁLISIS DE MÉTODOS DE REDUCCIÓN DE VARIANZA Y SU IMPLEMENTACIÓN AL CÓDIGO MONTE CARLO OPENMC

Debárbora Mauricio Ezequiel

Dr. José Ignacio Márquez Damián
Director
Mgter. Marquez Ariel
Co-director

Miembros del Jurado
Dr. Edmundo Lopasso (Instituto Balseiro)
Ing. Daniel Hergenreder (INVAP)

18 de julio de 2020

Departamento Física de Neutrones
División Física de Reactores Avanzados
Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

A mi familia

Índice de contenidos

Índice de contenidos ii

Índice de figuras iv

Índice de tablas viii

Índice de símbolos ix

Resumen x

Abstract xi

1. Introducción 1

1.1. OpenMC . . . 1

1.1.1. API . . . 2

1.1.2. Código fuente . . . 3

1.2. Tallies . . . 5

1.2.1. Estadística del tally con partículas sin peso asociado . . . 6

1.2.2. Estadística del tally con partículas con peso asociado . . . 7

1.2.3. Comparación entre Var $(\hat{x})$ y Var $(\hat{x}_w)$ . . . 8

1.3. Reducción de varianza . . . 10

1.3.1. Flujo adjunto y mapa de importancias . . . 11

2. Validación del código OpenMC 13

2.1. Benchmark de criticidad . . . 13

2.2. Benchmark de transporte de neutrones . . . 15

2.3. Verificación de la física de fotones . . . 17

3. Métodos de reducción de varianza 20

3.1. Métodos de reducción de varianza . . . 20

3.1.1. Figura de mérito *FOM* . . . 21

3.2. Captura implícita o survival biasing . . . 21

3.3. Geometry splitting y ruleta rusa . . . 21

3.3.1. Implementación del código . . . 25
3.3.2. Ruleta rusa como control poblacional . . . 26
3.4. Weight window . . . 27
3.4.1. Implementación del código . . . 28

4. Mapeo de importancias y generador de ventanas 30
4.1. Introducción . . . 30
4.2. Generador de puntos . . . 32
4.2.1. Clasificación de los puntos . . . 33
4.3. Estimación de importancia . . . 34
4.3.1. Proceso de cálculo . . . 35
4.3.2. Secciones eficaces . . . 38
4.3.3. Escaleo de la importancia . . . 39
4.4. Generador de ventanas . . . 39
4.5. Análisis del tiempo del mapeo de importancias . . . 39
4.6. Input del generador de mapa de importancias y de ventanas . . . 42

5. Aplicación de los métodos de reducción de varianza 44
5.1. Problema ejemplo para neutrones . . . 44
5.1.1. Geometría . . . 44
5.1.2. Mapeo de las importancias . . . 45
5.1.3. Resultados . . . 46
5.2. Benchmark numérico de fotones . . . 51
5.2.1. Datos del benchmark . . . 51
5.2.2. Resultados . . . 53
5.2.3. Mapeo de importancias de la geometría . . . 58

6. Conclusiones 64

A. Demostraciones 66
A.1. Prueba de la varianza de una combinación lineal . . . 66

B. Códigos utilizados 67
B.1. Códigos implementados para geometry splitting . . . 67
B.2. Código implementado de la ruleta rusa para control poblacional . . . 69
B.3. Códigos implementados para weight window . . . 69
B.4. Input de MCNP para el benchmark de fotones . . . 70

Bibliografía 73

Agradecimientos 75

Índice de figuras

1.1. Diagrama de flujo de interacción entre el *API* y el código fuente en la etapa de preprocesamiento de datos. 2
1.2. Diagrama de flujo del funcionamiento general del *código fuente* de *OpenMC*. Ver figura 1.3 para observar como se lleva a cabo el transporte de cada una de las partículas. 4
1.3. Diagrama de flujo del transporte de cada partícula. 5
1.4. Esquema representativo de la conservación de los pesos de las partículas. En este caso $n_i = 2$, es decir que la partícula i se dividió en 2. 7

2.1. Geometría utilizada para el benchmark de criticidad del *ZPR-6/6A*. Imagen tomada de [1]. 14
2.2. Distribución en energías de la fuente de $D - T$. 15
2.3. Geometría utilizada para el benchmark de fuente fija como se describe en la Ref. 2. Todas las dimensiones se encuentran en cm. 16
2.4. Espectros de neutrones obtenidos mediante la simulación del benchmark utilizando MCNP y OpenMC. También se muestra el espectro medido experimentalmente. Los espectros se encuentran normalizados por unidad de letargía. 16
2.5. Espectro del flujo promedio en el volumen de la esfera utilizando los códigos de cálculo OpenMC y MCNP. 18
2.6. Diferencia relativa del espectro obtenido con OpenMC respecto del calculado con MCNP. 18

3.1. Esquema representativo de un haz de partículas incidentes sobre un blindaje, en el cual se coloca un tally en el extremo opuesto al haz. 20
3.2. Esquema de Geometry splitting. Este método busca que la cantidad de partículas aumente en la dirección en la cual aumenta la importancia y que disminuya en el caso contrario. 22
3.3. Esquema representativo de la división de una partícula al entrar a una celda de mayor importancia. 23

3.4. Esquema representativo de la división de una partícula al entrar a una celda de menor importancia. 24
3.5. Esquema representativo del funcionamiento del método de reducción de varianza *Weight window*. Se puede ver que una partícula con peso superior al de la ventana se divide en otras con un peso tal que estas se encuentran dentro de la ventana. Si la partícula tiene un peso inferior al de la ventana esta puede terminar su historia (representada con "poof" en el gráfico) o terminar con un peso que se encuentre dentro de la ventana. Imagen tomada de [3]. 27

4.1. Diagrama de flujo del funcionamiento del generador de mapa de importancias. 31
4.2. Corte axial de la geometría utilizada para pruebas dada por un cilindro y una esfera centrados en el origen de coordenadas. Las dimensiones están en cm. 32
4.3. Nube de puntos con la definición de la geometría del problema. 33
4.4. Nube de puntos correspondientes a la esfera que rodea al cilindro central. 34
4.5. Nube de puntos correspondientes al cilindro central. 34
4.6. Esquema representativo del proceso llevado a cabo para determinar le importancia del punto $P2$ respecto del punto $P1$. 36
4.7. Evolución de la importancia con la distancia recorrida. 36
4.8. Combinación de puntos iniciales. El código itera sobre una variable i la cual va desde $i = 1$ hasta $i = N$. Para cada i se cuenta con una combinación de punto inicial y punto final las cuales definen una importancia. Para el caso analizado en donde $N > n$, se tiene que cuando $i = n + 1$ se vuelve a tomar el punto $j = 1$ de la celda del tally. 37
4.9. Geometría utilizada para llevar a cabo el análisis del tiempo de cálculo del mapa de importancias. 40
4.10. Tiempo de ejecución del generador de mapa de importancias en función de la cantidad de partículas iniciales de la nube de puntos. 41
4.11. Importancia obtenida para cada celda en función del tiempo total de cálculo. 42

5.1. Geometría utilizada en el benchmark neutrónico. Cada celda tiene un forma cilíndrica y se pueden observar 3 celdas que resultan de mayor interés, las cuales se corresponden a la celda de la fuente puntual la cual se encuentra en el origen de coordenadas y las otras dos son tallies volumétrico. La unidad de todas las cotas es cm. 45

5.2. Espectros de flujo de neutrones utilizando los distintos métodos de reducción de varianza para el caso del tally de la celda no importante. También se presentan los espectros de flujo de neutrones obtenidos sin utilizar técnicas de reducción de varianza para ambos tallies. 47
5.3. Error relativo de los espectros de flujo de neutrones presentados en la figura 5.2. 47
5.4. Espectro de neutrones resultante en el tally de flujo neutrónico en el volumen de la celda 2 para los distintos modos de simulación utilizados. 48
5.5. Error relativo del espectro de neutrones en el tally de flujo neutrónico de la celda importante para los distintos métodos de reducción de varianza utilizados. 49
5.6. Error relativo en el pico de la Maxwelliana para distintas cantidades de partículas en función del tiempo total de cálculo de los mismos. Para el caso en el cual se aplicaron los métodos de reducción de varianza, el tiempo total de cálculo es la suma del tiempo de simulación de OpenMC, el tiempo empleado por el generador de mapa de importancias y el tiempo utilizado para la generación de las secciones eficaces que usa el generador. Por otro lado, para el caso sin reducción de varianza el tiempo total de cálculo es solamente el tiempo de simulación de OpenMC. 50
5.7. *FOM* tomando como error relativo en el pico de la Maxwelliana en función de la cantidad de partículas de fuente simuladas. El tiempo empleado para el cálculo de la FOM es el tiempo total de cálculo que se encuentra en la figura 5.6. 50
5.8. Geometría dada por el benchmark de fotones. Las dimensiones se encuentran en cm. 51
5.9. Coeficientes de dosis ambientales equivalentes para fotones. Se puede observar que los fotones de más alta energía son aquellos que más aportan a la dosis total. 52
5.10. Espectro del flujo de fotones obtenido para las simulaciones con OpenMC y MCNP. 54
5.11. Error relativo del espectro del flujo de fotones obtenido para las simulaciones con OpenMC y MCNP. 54
5.12. Espectro del flujo de fotones con los errores obtenidos utilizando ambos códigos de cálculo sin métodos para reducir la varianza. 55
5.13. Error relativo del espectro del flujo de fotones a 1MeV en función del tiempo de cálculo empleado para distinta cantidad de partículas. 56
5.14. Figura de mérito en función de la cantidad de partículas simuladas. Se observa un aumento para ambos métodos de reducción de varianza aplicados. 56

5.15. Error relativo de la dosis total en función del tiempo de cálculo empleado por OpenMC. 57
5.16. Figura de mérito de la dosis total en función de la cantidad de partículas simuladas. Se observa un aumento para ambos métodos de reducción de varianza aplicados. 58
5.17. Espectros del flujo de fotones obtenido para las simulaciones con OpenMC utilizando el mapa de importancias generado. 59
5.18. Error relativo de los espectros del flujo de fotones obtenido para las simulaciones con OpenMC utilizando el mapa de importancias generado. 59
5.19. Comparación de los errores relativos obtenidos para el caso en el cual se utiliza el generador de mapa de importancias respecto de la asignación manual del mismo. 60
5.20. Error relativo del espectro del flujo de fotones a 1MeV en función del tiempo de cálculo empleado para distintas cantidades de partículas. El tiempo total de cálculo representa la suma del tiempo empleado por el generador de mapa de importancias, el tiempo utilizado para calcular las secciones eficaces y el tiempo de simulación de OpenMC, para el caso en el cual se aplicaron métodos de reducción de varianza. Por otro lado, para el caso sin reducción de varianza el tiempo total de cálculo es solamente el tiempo de simulación de OpenMC. 61
5.21. Figura de mérito en función de la cantidad de partículas simuladas. El tiempo empleado para el cálculo de la FOM es el tiempo total de cálculo que se encuentra en la figura 5.20. 61
5.22. Error relativo de la dosis total en función del tiempo total de cálculo. . 62
5.23. Figura de mérito en función de la cantidad de partículas de fuente simuladas. 63

Índice de tablas

2.1. En esta tabla se presenta el resultado experimental, el calculado con MCNP y de la simulación llevada a cabo con OpenMC. 14

5.1. Mapa de importancias generado mediante el código implementado en el *API* de OpenMC. 46
5.2. Mapa de importancias utilizados para la simulación del problema. . . . 53
5.3. Dosis total obtenidas para ambos códigos de transporte mediante método Monte Carlo y los distintos métodos de reducción de varianza aplicados. 57
5.4. Mapa de importancias obtenidos mediante la utilización del generador de importancias. 58
5.5. Dosis total y el error relativo de la misma obtenidos utilizando OpenMC y los distintos métodos de reducción de varianza. 62

Índice de símbolos

x_i: *i-ésima* observación.

w_i: Peso de la *i-ésima* observación.

$\bar{x}$: Valor medio de la variable x.

$\hat{x}$: Estimador de la media poblacional.

$\psi(\vec{r}, \Omega, E)$: Flujo angular.

Var(x_i): Varianza de la variable x_i.

σ^2: Varianza de la muestra.

$s_{\bar{x}}{}^2$: Varianza poblacional.

$\mathbf{H}$: Operador de transporte.

$\mathbf{H}^\dagger$: Operador de transporte adjunto.

ψ: Flujo.

$\psi^\dagger$: Flujo adjunto.

ψ_0: Función de Green solución de la ecuación de transporte.

Σ_d: Función de probabilidad de que los neutrones de la fuente q colisionen en una parte del espacio de fases.

Σ_{tr}: Sección eficaz macroscópica de transporte.

Σ_t: Sección eficaz macroscópica total.

Σ_a: Sección eficaz macroscópica de absorción.

Resumen

En el presente trabajo se lleva a cabo el análisis de métodos de reducción de varianza para acelerar cálculos de transporte de radiación por el método Monte Carlo. En particular, se estudiaron dos métodos de control de población: distribuciones de importancia (geometry splitting) y ventanas de peso (weight windows). Estos métodos fueron implementados en el código Monte Carlo OpenMC.

Como parte del trabajo se validó el código con tres benchmarks, uno de criticidad, uno de fuente fija de neutrones y el último de fuente fija de fotones, para luego implementar los métodos de reducción de varianza tanto en el código fuente en C++ como en el API en Python. Para simplificar el uso por parte del usuario se desarrolló una herramienta de cálculo de mapas de importancia y ventanas de peso, que genera en forma automática los parámetros necesarios para aplicar estos métodos de reducción de varianza.

La implementación de los métodos fue verificada con dos problemas de transporte de fuente fija, uno de transporte de neutrones y el otro de fotones, observándose un aumento en la velocidad de cálculo sin afectar el resultado.

Palabras clave: MÉTODO MONTE CARLO, OPENMC, REDUCCIÓN DE VARIANZA, GEOMETRY SPLITTING, WEIGHT WINDOW

Abstract

In the present work the analysis of the variance reduction techniques to accelerate radiation transport calculations by Monte Carlo method are presented. Particularly, two methods of population control were studied: Geometry splitting and Weight windows. These methods were implemented in the Monte Carlo code OpenMC.

Performing validation of the program has been made for both fixed source (photon and neutron) and critical problems simulations, and then the two methods were implemented in the source code in C++ and in the API in Python. To simplify the use by the user an importance and weight windows map generator was developed, wich generates the parameters needed by the two variance reduction techniques.

The correct operation of the variance reduction techniques and the importance map generator for the transport of neutrons and photons were verified. This was done by analyzing simple benchmarks for each type of particle. It was observed that the velocity of the program increase without affecting the result.

Keywords: MONTE CARLO METHOD, OPENMC, VARIANCE REDUCTION, GEOMETRY SPLITTING, WEIGHT WINDOW

Capítulo 1

Introducción

1.1. OpenMC

OpenMC es un código abierto de simulación mediante método Monte Carlo de transporte de neutrones y fotones, desarrollado inicialmente por el departamento de física de reactores del Instituto de tecnología de Massachusetts (MIT) [4]. Este código es capaz de realizar cálculos de fuente fija y auto valores (cálculos de criticidad).

Es muy importante el hecho de que el código sea open source ya que se cuentan con las cuatro libertades que tienen los software libre, las cuales son [5]:

0. La libertad de usar el programa, con cualquier propósito (uso).
1. La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a las propias necesidades (estudio).
2. La libertad de distribuir copias del programa, con lo cual se puede ayudar a otros usuarios (distribución).
3. La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie (mejora).

El código fuente está programado en *C++* y es un código que al haber sido desarrollado por completo recientemente, se encuentra bien documentado y programado en un alto nivel para optimizar al mismo. Por otro lado, cuenta con un *API* (Application Programming Interface) que fue desarrollado en *Python*, el cual se utiliza para el preprocesamiento llevado a cabo para generar los inputs de la simulación y el post-procesamiento de los resultados obtenidos.

Para el presente trabajo se utilizó la versión 0.12.0 de *OpenMC* en la versión de desarrollador, la cual permite acceder y modificar tanto el *API* como el *código fuente*. La biblioteca de secciones eficaces utilizada es *ENDF/B-VII.1* la cual puede obtenerse de la página de *OpenMC* [6].

El programa con todas las modificaciones realizadas sobre el *API* y el *código fuente* puede encontrarse en *GitHub* [7].

1.1.1. API

El input del código fuente se debe especificar mediante el uso de archivos de extensión `.xml` el cual se utiliza para el almacenamiento de datos. Estos archivos pueden ser generados manualmente siguiendo la forma exacta en la cual deben ser representados los datos en el input para que el código fuente lo lea correctamente.

Esto también puede llevarse a cabo mediante la utilización del *API*, basado en el lenguaje de programación *Python*, el cual es una herramienta diseñada específicamente para realizar este trabajo, lo cual hace que generar estos archivos sea mucho más sencillo, intuitivo y disminuye la probabilidad de equivocarse al crearlo. De esta forma se define la primer función general del *API* la cual consiste en el preprocesamiento de los datos para luego ser entregados al *código fuente*.

Los archivos denominados `materials.xml`, `geometry.xml` y `settings.xml`, correspondientes a los materiales [8], la geometría [9] y la configuración o setting [10] de la simulación respectivamente, son los inputs mínimos y necesarios para el funcionamiento del programa. También se pueden utilizar tallies, los cuales son especificados en un archivo `tallies.xml` o realizar un gráfico resultante en un archivo `plot.xml`.

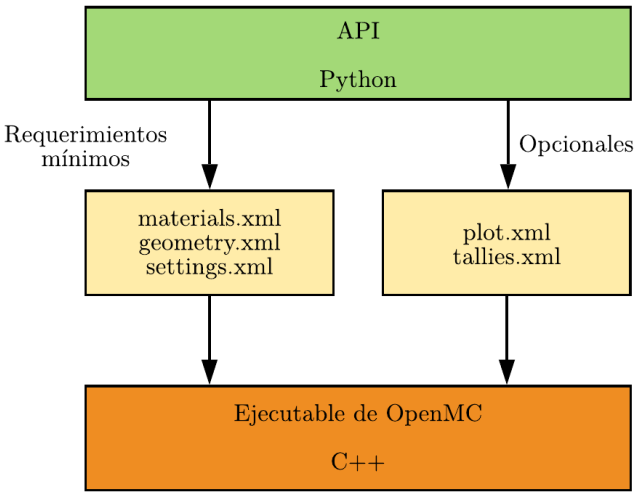

Figura 1.1: Diagrama de flujo de interacción entre el *API* y el código fuente en la etapa de preprocesamiento de datos.

Al especificar los materiales, se debe precisar la biblioteca de secciones eficaces microscópicas a utilizar en formato tipo HDF5, el cual es el tipo de archivos que maneja *OpenMC*. Estas son generadas desde el formato ENDF, las que mediante el uso de NJOY son transformadas al formato ACE y luego se puede utilizar un módulo del *API* de *Python* para transformar a éstos en archivos HDF5 [6].

El *API* cuenta con distintos módulos utilizados tanto para el preprocesamiento como para el postprocesamiento de datos [11]. Entre estos módulos, son de particular interés los siguientes:

- Funciones básicas: Generación de inputs como ser la geometría, materiales, settings, etc. Se lo utiliza también para el postprocesamiento de los resultados obtenidos.
- Generador de secciones eficaces multigrupo.
- Modulo *lib* el cual se encarga de ejecutar funciones definidas sobre el *código fuente* desde el *API*.

1.1.2. Código fuente

En la figura 1.2 se presenta un diagrama de flujo donde se muestra el funcionamiento general del código fuente, el cual es el encargado de llevar a cabo la simulación del transporte de partículas propiamente dicho.

Además, es de particular interés conocer de forma general como lleva a cabo el transporte de cada una de las partículas para la implementación de los métodos de reducción de varianza. Esto se debe a que para implementar cualquier modificación sobre el mismo, se debe comprender que pasos lleva a cabo el código para entender en qué parte incluir la misma.

Si bien el marco teórico no difiere del presentado en textos de cálculo neutrónico [12, 13], la implementación utilizando programación orientada objetos en OpenMC es bastante particular. Es por ello por lo cual gran parte del trabajo realizado fue entender y analizar el funcionamiento de este código.

En la figura 1.3 se puede ver como se lleva a cabo el transporte de cada partícula por *OpenMC* de forma general.

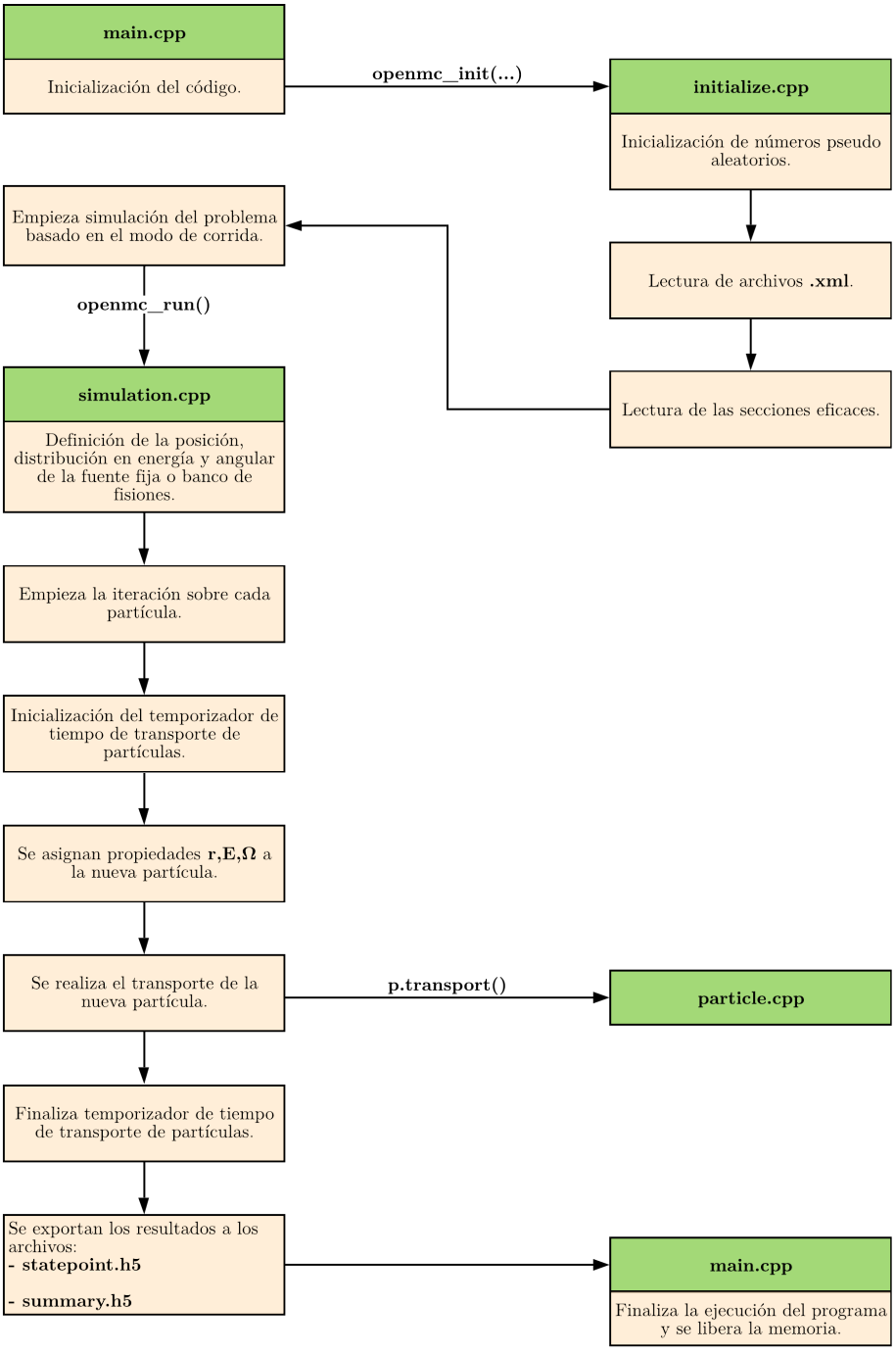

Figura 1.2: Diagrama de flujo del funcionamiento general del *código fuente* de *OpenMC*. Ver figura 1.3 para observar como se lleva a cabo el transporte de cada una de las partículas.

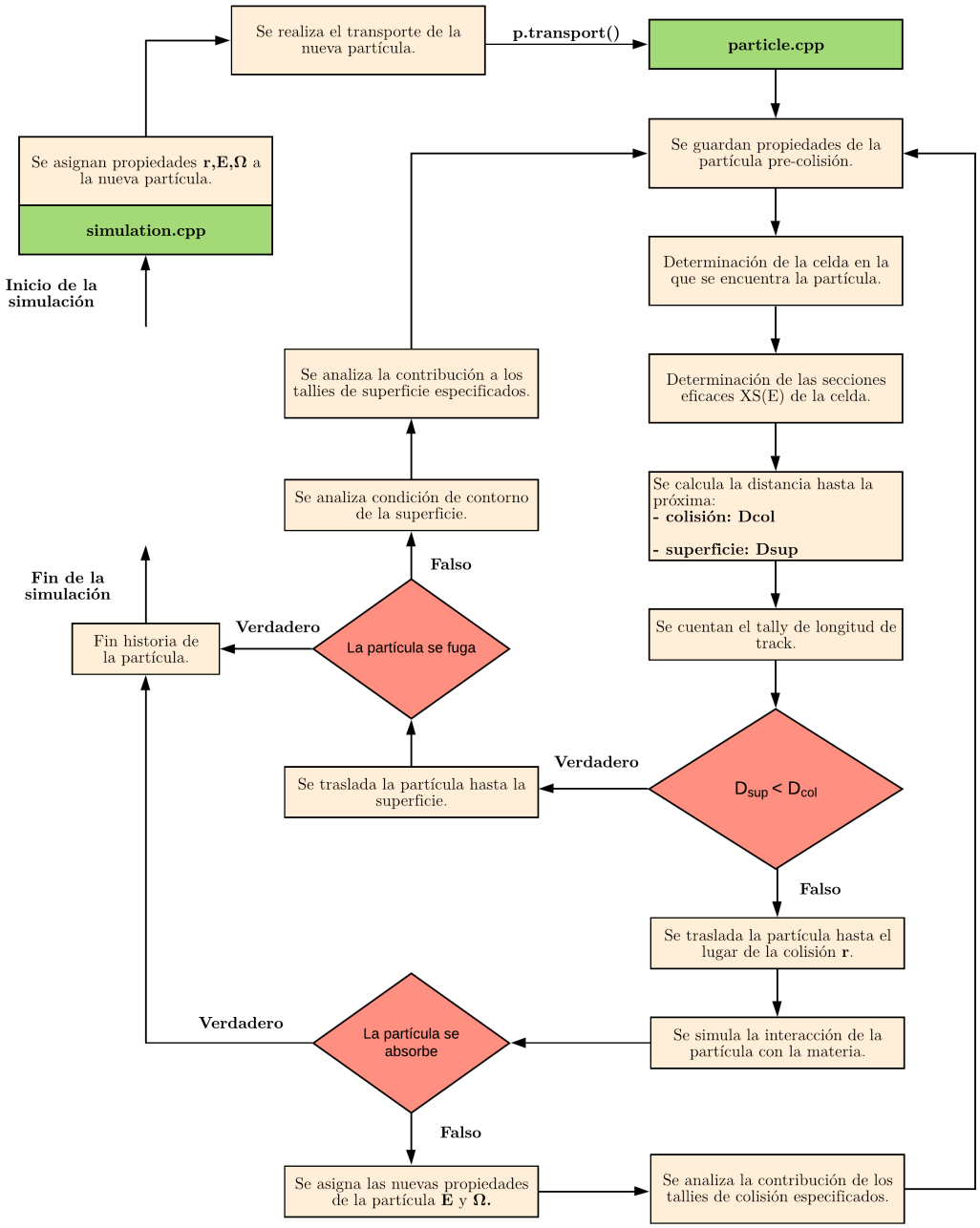

Figura 1.3: Diagrama de flujo del transporte de cada partícula.

1.2. Tallies

La determinación de una cantidad física en transporte de partículas mediante método Monte Carlo como ser el flujo neutrónico o ritmos de reacción, se lleva a cabo mediante lo que se denomina *tally*, que es un contador de una cantidad sobre cierto dominio del espacio de fases $\Delta V, \Delta E, \Delta\Omega$. Un contador o "tally" entonces puede expresarse en forma genérica como:

$$\hat{x} = \int_{\Delta V} dV \int_{\Delta E} dE \int_{\Delta \Omega} d\Omega \overbrace{f(\vec{r}, \Omega, E)}^{\text{Score}} \underbrace{\psi(\vec{r}, \Omega, E)}_{\text{Flujo angular.}} \tag{1.1}$$

donde $\hat{x}$ es la variable a estimar y el score es el factor que altera al flujo angular resultando en la cantidad que se quiere medir. En particular para el caso en el cual se quiera estimar un ritmo de reacción el score es la sección eficaz.

1.2.1. Estadística del tally con partículas sin peso asociado

Debido a que la variable que se va a contar dentro del dominio del espacio de fases es discreta podemos escribir al estimador de la media poblacional como el valor medio de las observaciones x_i.

$$\hat{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1.2}$$

donde N es la cantidad de partículas que se observan en el dominio y x_i es el valor de la observación i.

La ley fuerte de los grandes números (strong law of large numbers) establece que si se realiza un experimento de forma repetida una gran cantidad de veces, el valor medio de los resultados es cercano al valor esperado [14]. Como en las simulaciones realizadas mediante método Monte Carlo la cantidad de partículas simuladas N es grande, la estimación dada por la ecuación 1.2 se cumple.

Se tiene entonces que la esperanza del estimador $\hat{x}$ es:

$$E\left[\hat{x}\right] = E\left[\frac{1}{N} \sum_{i=1}^{N} x_i\right] = \frac{1}{N} \sum_{i=1}^{N} E\left[x_i\right] = \bar{x} \tag{1.3}$$

La varianza muestral se puede calcular utilizando como estimador el momento de segundo orden alrededor de la media. Teniendo en cuenta que las observaciones son independientes entre sí, la varianza de la muestra insesgada es [15]:

$$\mathrm{Var}(x_i) = \sigma^2 = \frac{1}{N-1} (\sum_{i=1}^{N} {x_i}^2 - N\bar{x}^2) \tag{1.4}$$

Pero como no estamos buscando la varianza muestral σ^2, sino la varianza $s_{\bar{x}}^2$ del estimador $\hat{x}$ del valor medio poblacional $\bar{x}$, podemos escribir a $s_{\bar{x}}^2$ en función de σ^2 como [12]:

$$\mathrm{Var}\left(\hat{x}\right) = \mathrm{Var}\left(\frac{1}{N} \sum_{i=1}^{N} x_i\right) = \frac{1}{N^2} \sum_{i=1}^{N} Var\left(x_i\right) = \frac{\sigma^2}{N} \tag{1.5}$$

Finalmente se obtiene que la varianza del estimador del valor medio poblacional es:

$$\mathrm{Var}(\hat{x}) = s_{\bar{x}}^2 = \frac{1}{N-1}(\frac{1}{N}\sum_{i=1}^{N} {x_i}^2 - \bar{x}^2) \tag{1.6}$$

1.2.2. Estadística del tally con partículas con peso asociado

Dada la simulación en *OpenMC* de la *i-ésima* partícula, se le atribuye a la misma un peso w_i donde $w_i > 0$ para todo i. Este peso se podría pensar como una propiedad de la partícula que permite transformar en un contínuo la propiedad discreta de existencia de la partícula y es muy útil para reducir la varianza del problema sin modificar la naturaleza del mismo.

Podemos reescribir al estimador utilizado anteriormente partiendo del hecho de que una partícula que antes aportaba con peso unitario al tally, ahora son n_i partículas de peso w_{ij} aportando al mismo, donde j es la *j-esima* partícula en la cual se dividió a x_i.

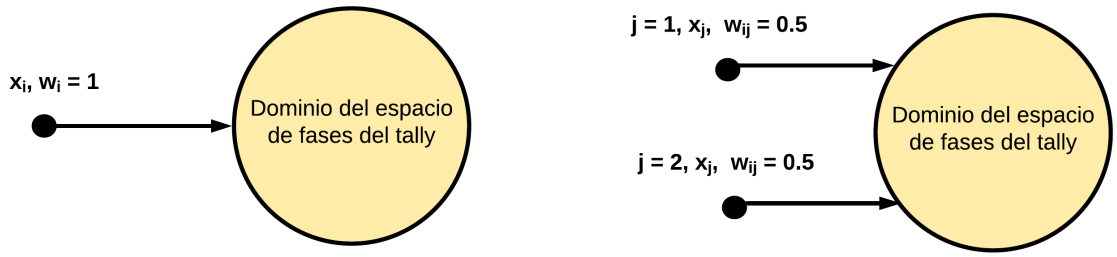

Figura 1.4: Esquema representativo de la conservación de los pesos de las partículas. En este caso $n_i = 2$, es decir que la partícula i se dividió en 2.

Se pretende utilizar el mismo estimador dado por 1.2 debido a que se busca mantener la esperanza del mismo para conservar el problema físico. Para ello se escribe a cada observación x_i como la suma de las observaciones x_j de las partículas en las cuales fue dividida. Además se sabe que cada observación x_j es la misma observación x_i afectado simplemente por el peso w_{ij}.

$$x_i = \sum_{j=1}^{n_i} x_j = \sum_{j=1}^{n_i} x_i w_{ij} \tag{1.7}$$

Como necesitamos que el peso se conserve, se tiene que:

$$\sum_{j=1}^{n_i} w_{ij} = 1 \tag{1.8}$$

La ecuación 1.2 entonces se puede escribir como:

$$\hat{x}_w = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n_i} x_i w_{ij} \tag{1.9}$$

Finalmente, definiendo $P = \sum_{i=1}^{N} n_i$ como la cantidad total de partículas de peso distinto de cero que aportan al tally se obtiene que el estimador del valor medio para partículas con pesos asociados es:

$$\hat{x}_w = \frac{\sum_{k=1}^{P} w_k x_k}{M} \tag{1.10}$$

en donde $w_k x_k = w_{ij} x_i$, $x_k = x_i$ y $w_k = w_{ij}$. Por otro lado M es:

$$M = \sum_{k=1}^{P} w_k \tag{1.11}$$

Se puede observar que como la sumatoria esta dada sobre los P pesos de partículas, se tiene que $M = N$. La esperanza del nuevo estimador $\hat{x}_w$ es:

$$E\left[\hat{x}_w\right] = E\left[\frac{1}{M}\sum_{i=1}^{P} w_k x_k\right] = E\left[\frac{1}{N}\sum_{i=1}^{N} x_i\right] = E\left[\hat{x}\right] = \bar{x} \tag{1.12}$$

En este caso se tiene que la esperanza de ambos estimadores son iguales a $\bar{x}$, es decir que se puede afirmar que se conserva la naturaleza del problema.

Para hacer un análisis del impacto que esto tiene sobre la varianza, empezamos analizando la varianza muestral de observaciones que tienen un peso asociado. Al igual que en 1.4, se tiene que para la variable $w_i x_i$ el estimador de su varianza muestral es:

$$\mathrm{Var}(w_i x_i) = {\sigma_w}^2 = \frac{1}{P-1}\left(\sum_{i=1}^{P}\left(w_k x_k\right)^2 - P\left(\bar{wx}\right)^2\right) \tag{1.13}$$

Donde $\bar{wx}$ es el valor medio de la variable $w_i x_i$, que es igual a:

$$\bar{wx} = \frac{1}{P}\sum_{k=1}^{P} w_k x_k \tag{1.14}$$

Por último, la varianza del estimador $\hat{x}_w$, es decir, la varianza poblacional será:

$$\mathrm{Var}\left(\hat{x}_w\right) = \mathrm{Var}\left(\frac{1}{M}\sum_{k=1}^{P} w_k x_k\right) = \frac{1}{M^2}\mathrm{Var}\left(\sum_{k=1}^{P} w_k x_k\right) = \frac{P}{M^2}{\sigma_w}^2 \tag{1.15}$$

1.2.3. Comparación entre $\mathrm{Var}\left(\hat{x}\right)$ y $\mathrm{Var}\left(\hat{x}_w\right)$

Ahora bien, la idea es comparar la varianza de los estimadores $\hat{x}$ y $\hat{x}_w$. Por lo visto anteriormente se tiene que:

$$\mathrm{Var}\left(\hat{x}_w\right) = \frac{1}{M^2}\,\mathrm{Var}\left(\sum_{k=1}^{P} w_k x_k\right)$$

Por otro lado, se puede escribir a la varianza de una combinación lineal como (ver apéndice A.1 para demostración):

$$\mathrm{Var}\left(\sum_{k=1}^{P} w_k x_k\right) = \sum_{k=1}^{P} w_k^2\,\mathrm{Var}(x_k) + 2\sum_{h=1}^{P}\sum_{l:\,l>h}^{P} w_h w_l\,\mathrm{Cov}(x_h, x_l) \tag{1.16}$$

Como las observaciones x_k son independientes entre sí se tiene que $\mathrm{cov}(x_h, x_l) = 0$, y además que $\mathrm{Var}(x_k) = \sigma^2$. Por lo tanto se puede expresar a la varianza del estimador como:

$$\mathrm{Var}\left(\hat{x}_w\right) = \frac{1}{M^2}\sum_{k=1}^{P}\mathrm{Var}(x_k)w_k^2 = \frac{1}{M^2}\sigma^2\sum_{k=1}^{P} w_k^2 \tag{1.17}$$

Planteando la condición para la cual $\mathrm{Var}\left(\hat{x}_w\right) < \mathrm{Var}\left(\hat{x}\right)$ se obtiene:

$$\begin{aligned}
\mathrm{Var}\left(\hat{x}_w\right) &< \mathrm{Var}\left(\hat{x}\right)\\
\frac{1}{M^2}\sigma^2\sum_{k=1}^{P} w_k^2 &< \frac{\sigma^2}{N}\\
\sum_{k=1}^{P} {w_k}^2 &< N\\
\sum_{i=1}^{N}\sum_{j=1}^{n_i} {w_{ij}}^2 &< \sum_{i=1}^{N} 1
\end{aligned}$$

De donde se puede concluir que para que $\mathrm{Var}\left(\hat{x}_w\right) < \mathrm{Var}\left(\hat{x}\right)$ se debe cumplir que:

$$\sum_{j=1}^{n_i} {w_{ij}}^2 < 1 \tag{1.18}$$

Como cada partícula se subdivide en n_i partículas de peso w_{ij} y producto de que el peso total de las partículas se conserva por 1.8 sabemos que $0 \leqslant w_{ij} < 1$. Por lo tanto se cumple que ${w_{ij}}^2 < w_{ij}$ y entonces partiendo de la ecuación 1.18 y 1.8:

$$\sum_{j=1}^{n_i} {w_{ij}}^2 < \sum_{j=1}^{n_i} w_{ij} = 1 \tag{1.19}$$

Esto quiere decir que la varianza de la simulación es menor para el caso en el cual se utiliza como estimador del valor medio de las observaciones a $\hat{x}_{iw}$ para el caso en

el cual el tally se encuentre en un dominio del espacio de fases en el cual se cumpla 1.8. Esta aclaración se debe a que, como se verá más adelante, las técnicas utilizadas para reducir la varianza de la simulación modifican el problema (respecto del caso en el cual no hay ningún método de reducción de varianza aplicado) de manera tal que en algunas regiones del espacio aumenta la cantidad de partículas disminuyendo su peso y en otras disminuye la cantidad de partículas aumentando el peso de éstas.

1.3. Reducción de varianza

Anteriormente se obtuvo para $\mathrm{Var}(w_i x_i)$ y $\mathrm{Var}\left(\hat{x}_w\right)$:

$$\mathrm{Var}(w_k x_k) = {\sigma_w}^2 = \frac{1}{P-1}\left(\sum_{k=1}^{P}\left(w_k x_k\right)^2 - P\left(\bar{wx}\right)^2\right)$$

$$\mathrm{Var}\left(\hat{x}_w\right) = \frac{1}{N^2}\,\mathrm{Var}\left(\sum_{i=1}^{P} w_i x_i\right) = \frac{P}{N^2}{\sigma_w}^2$$

Se puede observar que para disminuir $\mathrm{Var}\left(\hat{x}_w\right)$ se pueden realizar diferentes acciones:

Aumento de *N*

La primer solución y la más fácil para disminuir la varianza del problema es aumentar la cantidad de partículas iniciales que se van a simular, debido a que ${\sigma_w}^2 \propto \frac{1}{N}$. Esta forma de bajar la desviación tiene como principal problema el hecho de que la cantidad de partículas simuladas es proporcional al tiempo T que tarda el código en realizar el transporte de las partículas.

Hallar flujo adjunto

Se puede calcular cada w_i tal que cada $w_i x_i = \bar{wx}$, pero esto conlleva a resolver la ecuación de transporte adjunto del problema, el cual es igualmente costoso que resolver la ecuación de transporte.

Disminución del peso de las partículas

De la ecuación 1.18 se puede ver que cuanto menor sea $\sum_{j=1}^{n_i} {w_{ij}}^2$, mayor va a ser la disminución de la varianza respecto del caso en el cual todas las partículas tienen peso unitario. El problema consiste en cómo hacer para disminuir el peso de las partículas sin modificar el problema físico de una forma eficiente, de manera que el tiempo de simulación sea utilizado en partículas que van a aportar al tally. De este problema se encargan los *métodos de reducción de varianza*, en el cual se adoptan modelos que producen un aumento en la eficiencia de la simulación.

1.3.1. Flujo adjunto y mapa de importancias

Partiendo de la definición del producto interno entre dos funciones g y f como la integral sobre todo el espacio de fases del producto entre ambas.

$$\langle g, f \rangle = \int g(\xi) f(\xi)\, d\xi \tag{1.20}$$

Entonces se define al operador adjunto $\mathbf{M}^\dagger$ con auto funciones $g^\dagger$, del operador $\mathbf{M}$ con auto funciones f como:

$$\langle g^\dagger, \mathbf{M} f \rangle = \langle \mathbf{M}^\dagger g^\dagger, f \rangle \tag{1.21}$$

Por otro lado, el operador de transporte $\mathbf{H}$ se define como:

$$\begin{aligned} \mathbf{H}(r, \Omega, E) = &-\Omega \cdot \nabla \psi(r, \Omega, E) - \sigma(r, E)\psi(r, \Omega, E) \\ &+ \iint \psi(r; \Omega', E') \sigma f(r; \Omega', E' \longrightarrow \Omega, E)\, d\Omega'\, dE' \end{aligned} \tag{1.22}$$

Es posible definir un operador adjunto $\mathbf{H}^\dagger$ al operador $\mathbf{H}$, en donde su auto función es el flujo adjunto $\psi^\dagger$ [16]. Por lo tanto se tiene que:

$$\langle \psi^\dagger, \mathbf{H}\psi \rangle = \langle \mathbf{H}^\dagger \psi^\dagger, \psi \rangle \tag{1.23}$$

Analizando el es caso en el cual se cuenta con una fuente arbitraria de neutrones $q(r, E, \Omega)$ en un sistema subcrítico de estado estacionario, se tiene que:

$$\mathbf{H}\psi = -q \tag{1.24}$$

Por el otro lado, se plantea para el operador de transporte adjunto:

$$\mathbf{H}^\dagger \psi^\dagger = -\sigma_d \tag{1.25}$$

en donde σ_d es la función de probabilidad correspondiente a que los neutrones de la fuente $q(r, E, \Omega)$ colisionen en una parte del espacio de fases. En particular es de interés analizar el caso en el cual esta función define la probabilidad de colisionar en la celda del tally, ya que es la zona de la geometría en la cual se busca que se lleve a cabo una interacción.

Utilizando estas dos últimas ecuaciones y la definición de operadores adjuntos se obtiene que [16]:

$$\int q(r, E, \Omega)\psi^\dagger(r, \Omega, E)\, dV\, d\Omega\, dE = \int \sigma_d(r, E)\psi(r, \Omega, E)\, dV\, d\Omega\, dE \tag{1.26}$$

El lado derecho de la ecuación expresa el ritmo de reacción dentro del tally producto de los neutrones de la fuente q. La ecuación 1.26 se cumple para cualquier fuente de neutrones q ya que esta fue definida de forma aleatoria. Esto se debe a que la definición 1.21 esta dada sobre ambos operadores, cada uno aplicados sobre su auto función, pero no así sobre la inhomogeneidad que representa la fuente en cada caso.

En particular, definiendo a q como una fuente puntual en el espacio de fases (r_0, E_0, Ω_0) se tiene que:

$$\psi^\dagger(r_0, \Omega_0, E_0) = \int \sigma_d \psi_0 \, dV \, d\Omega \, dE \tag{1.27}$$

donde ψ_0 es la función de Green solución de la ecuación de transporte. Entonces se tiene que $\psi^\dagger$ es proporcional al ritmo de reacción en el tally dado por el lado derecho de la ecuación 1.27. De esta forma se puede decir que el flujo adjunto $\psi^\dagger$ mide la "importancia" de que los neutrones de la fuente q contribuyan al tally.

Debido a que conocer el flujo adjunto permite tener una varianza cero, se pretende, en los programas de transporte de partículas mediante método Monte Carlo tratar de obtener valores aproximados de la importancia que tiene cada celda, de manera en la cual se pueda tener una distribución de pesos que tienda a disminuir la varianza del tally.

En otras palabras, se busca simular la mayor cantidad de partículas que se encuentren en un punto del espacio de fases que tenga una importancia alta respecto del tally, porque la probabilidad de que estas aporten al mismo son mayores que aquellas partículas que estén un punto de menor importancia.

Como se mencionó anteriormente, conocer el mapa de importancias del problema conllevaría entonces a tener una solución con varianza cero. El problema radica en que obtener el flujo adjunto para cada punto del espacio de fases es igual de costoso computacionalmente que llevar a cabo el análisis del flujo. Es por ello por lo cual existen técnicas de reducción de varianza que utilizan a la ecuación de transporte adjunta y buscan aproximarla de alguna manera razonable.

Capítulo 2

Validación del código OpenMC

Para comenzar a utilizar el código OpenMC es necesario realizar un proceso de validación. Si bien las definiciones precisas dependen del campo de aplicación, el control de calidad del software requiere de un proceso de verificación y validación [17]. *Verificar* el código implica realizar pruebas para asegurar que los algoritmos están implementados en forma correcta y funcionan correctamente, mientras que *validar* el código (y en caso de neutrónica, también las secciones eficaces y el usuario) implica simular problemas *benchmark* para comparar los cálculos con resultados experimentales.

La verificación del código se realizó corriendo los tests implementados al compilarlo. Por otra parte, para validar el conjunto usuario, código, secciones eficaces, se analizaron tres benchmarks experimentales: uno de criticidad y dos de fuente fija (uno de neutrones y otro de fotones).

2.1. Benchmark de criticidad

Para llevar a cabo la validación de OpenMC para cálculos de criticidad se simuló el benchmark IEU-COMP-FAST-001 del *International Criticiality Safety Benchmark Evaluation Project* [1]. El mismo se denomina *ZPR-6/6A* y se corresponde con un reactor de potencia cero rápido refrigerado por sodio.

Se cuenta con mediciones experimentales del k_{eff}, como así también calculado con otros códigos de transporte de partículas por método Monte Carlo como ser MCNP.

Se llevo a cabo el cálculo del k_{eff} mediante el código de cálculo OpenMC para después compararlo con los resultados reportados en el benchmark.

La geometría utilizada para llevar a cabo la simulación es la misma presentada en el informe de benchmark, el cual es un sistema simplificado en geometría cilíndrica y se puede observar en la siguiente figura:

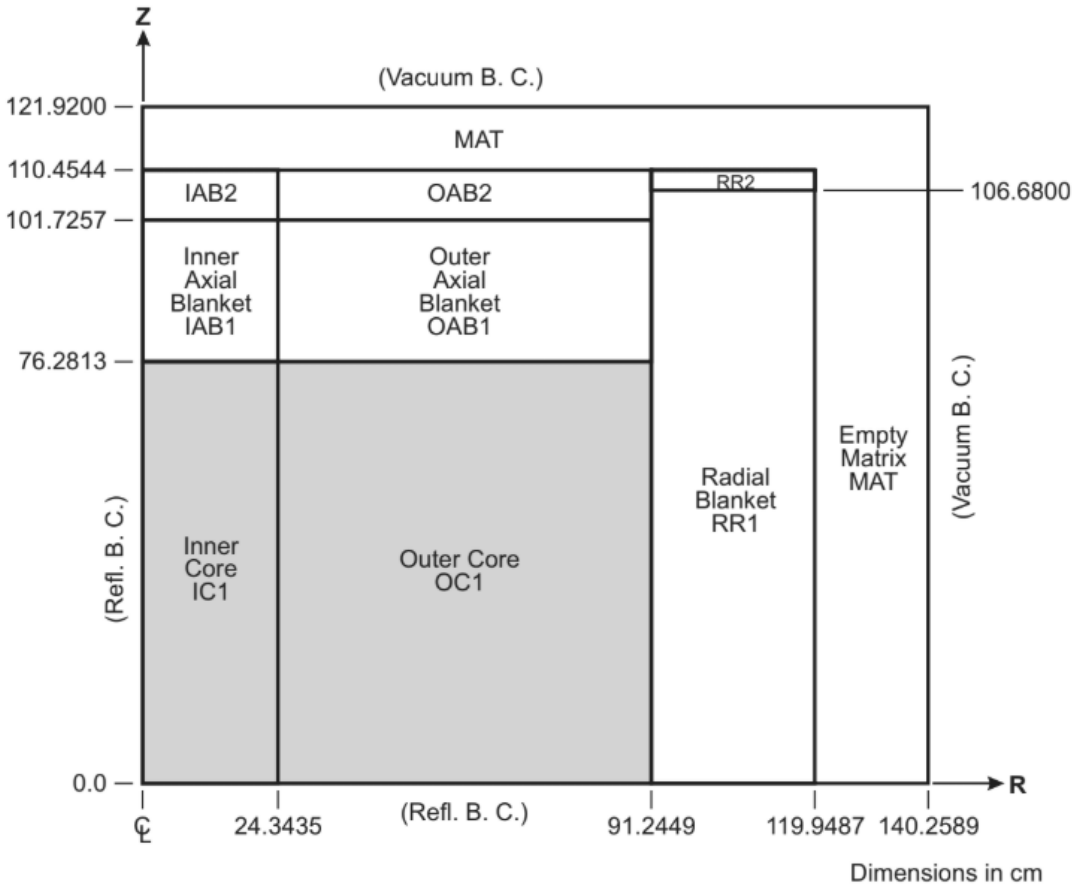

Figura 2.1: Geometría utilizada para el benchmark de criticidad del *ZPR-6/6A*. Imagen tomada de [1].

Los materiales utilizados son aquellos presentados en el informe del benchmark, en donde las regiones IC1 y OC1 corresponden al núcleo, IAB1 e IAB2 a los blankets internos, OAB1 y OAB2 a los blankets externos, RR1 y RR2 a los reflectores radiales, y MAT a la matriz externa del ZPR-6/6A.

En la tabla 2.1 se presenta el resultado experimental, el reportado en el informe en el cual se utilizó MCNP y por último el obtenido mediante la utilización del código de cálculo OpenMC.

	Benchmark	**MCNP5 (ENDF/B-VII β3)**	**OpenMC(ENDF/B-VII.1)**
keff $\pm$ σ	0,9939±0,0023	0,9931±0,0001	0,9931±0,0001
(C-E)/E[%]	-	-0,08 $\pm$ 0,23	-0,08 $\pm$ 0,23

Tabla 2.1: En esta tabla se presenta el resultado experimental, el calculado con MCNP y de la simulación llevada a cabo con OpenMC.

Se puede observar que los resultados obtenidos utilizando OpenMC y MCNP son idénticos, y que la diferencia porcentual respecto del valor experimental de k_{eff} difiere en una cantidad menor al 1 %. De esta manera se validó a OpenMC para cálculos de criticidad.

2.2. Benchmark de transporte de neutrones

Para llevar a cabo la validación de OpenMC para cálculos de fuente fija se simuló un benchmark que consiste en medir el espectro de los neutrones que se fugan de una esfera de hierro de 50,32 cm de radio, sobre los cuales se cuentan con los resultados experimentales del mismo. El informe del benchmark utilizado es [2].

La fuente de neutrones es una fuente de 14MeV de $D-T$ perteneciente a una de las facilidades denominada *Oktavian* de la Universidad de Osaka, Japón. En la figura 2.2 se puede observar el espectro de la fuente de neutrones utilizada.

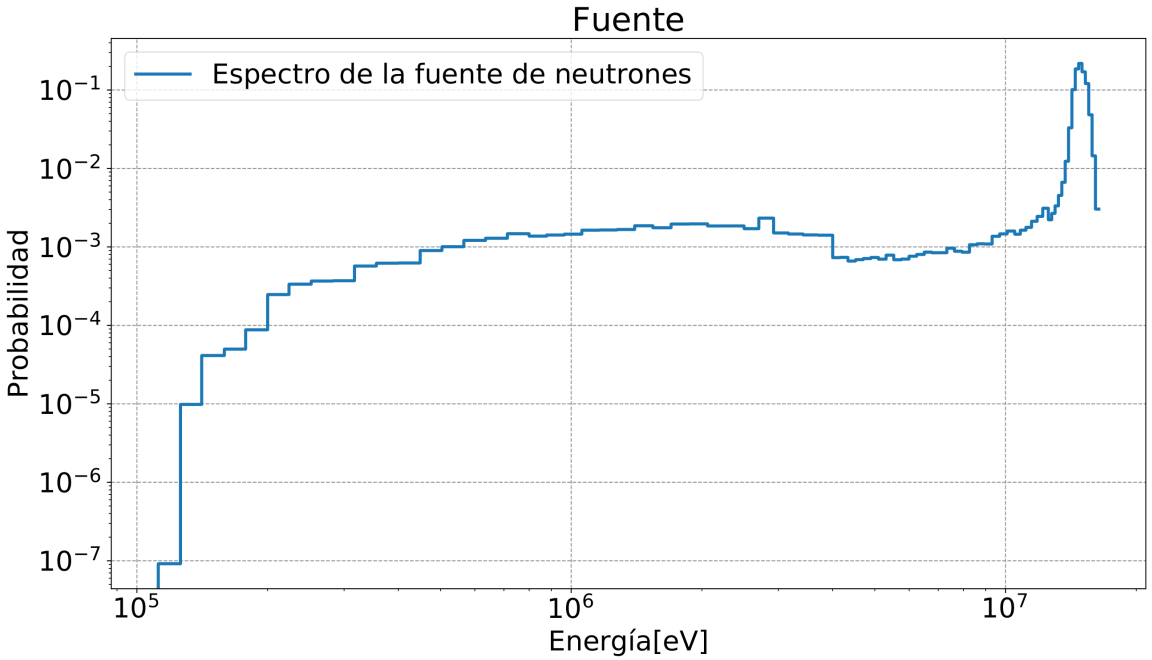

Figura 2.2: Distribución en energías de la fuente de $D-T$.

La geometría utilizada para la simulación del mismo se puede observar en la figura 2.3. Debido a que la especificación de la configuración geométrica no es precisa en el informe del benchmark, las medidas utilizadas provienen de un análisis de una imagen de la configuración experimental dada en el informe.

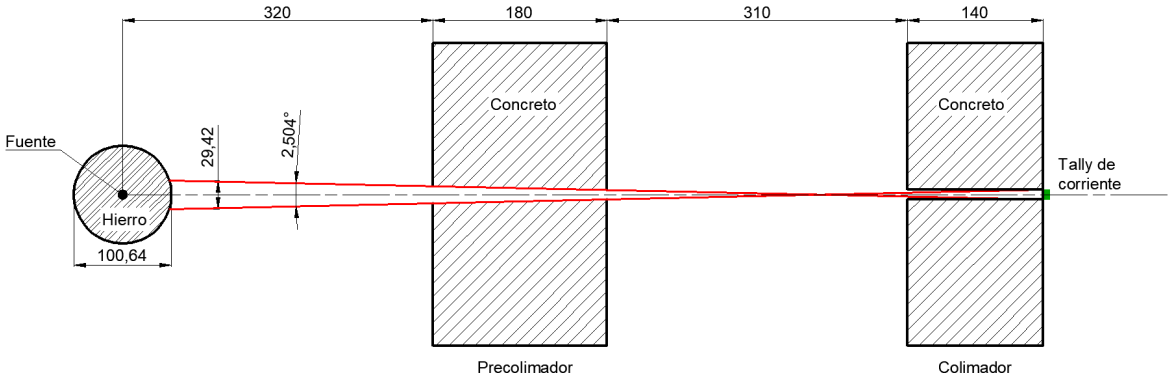

Figura 2.3: Geometría utilizada para el benchmark de fuente fija como se describe en la Ref. 2. Todas las dimensiones se encuentran en cm.

Se puede observar que se cuenta con un precolimador y un colimador, y al final un detector de neutrones. Los dos primeros son utilizados para seleccionar los neutrones de un determinado ángulo sólido.

Utilizando OpenMC se simuló el benchmark utilizando una fuente con la distribución que se muestra en la figura 2.2, la geometría recién señalada, los materiales presentados en el informe del benchmark, un tally de corriente en el lugar donde se encuentra el detector y una cantidad de partículas de fuente de 5×10^8. En la figura 2.4 se observan los espectros de neutrones obtenidos con ambos códigos de cálculo, como así también el espectro medido experimentalmente. También se llevó a cabo la simulación con MCNP para llevar a cabo una comparación entre los distintos códigos de cálculo.

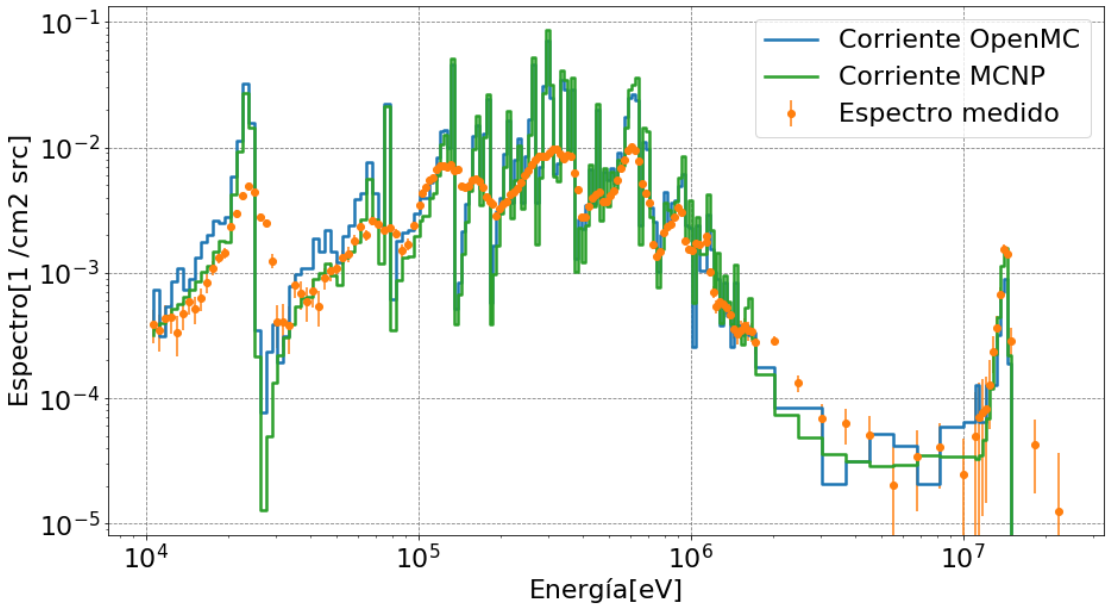

Figura 2.4: Espectros de neutrones obtenidos mediante la simulación del benchmark utilizando MCNP y OpenMC. También se muestra el espectro medido experimentalmente. Los espectros se encuentran normalizados por unidad de letargía.

Se verifica que los espectros calculados con MCNP y OpenMC presentan la misma forma funcional. Por otro lado, también se observa que el espectro medido experimentalmente concuerda con aquellos obtenidos mediante simulaciones. De esta forma se comprueba el correcto funcionamiento del código OpenMC para simulación de problemas con fuente fija de neutrones.

Por otro lado, en la figura 2.4 se puede observar que el corte a altas energías del espectro medido y los calculados con ambos códigos difieren entre si. Esto se debe al ensanchamiento por resolución experimental del benchmark, lo que produce también el ensanchamiento de los picos en la región de 1×10^4 a 1×10^6 eV. En [18] se encuentra un análisis mas detallado del ensanchamiento por resolución dado en el experimento.

2.3. Verificación de la física de fotones

Para llevar a cabo la verificación de la física de los fotones se modeló un problema benchmark planteado por Amanda Lund y Paul Romano en "Implementación y validación del transporte de fotones en OpenMC" ANL/MCS-TM-381 del *Argonne National Laboratory* [19], en el cual se compara el resultado obtenido con MCNP. El mismo consiste en una esfera de *Fe* natural con una densidad de 7,874 $\frac{\text{g}}{\text{cm}^3}$, con una fuente puntual de fotones de 10MeV en el centro de la misma. Se utiliza un tally de flujo promedio en el volumen de la esfera.

En la figura 2.5 se puede observar el flujo de fotones obtenido utilizando OpenMC y MCNP para la configuración recién descrita. En la figura 2.6 se muestra la diferencia relativa del espectro obtenido con OpenMC respecto del realizado con MCNP de la misma manera en la cual es presentado en el informe técnico. En ambas simulaciones realizadas se utilizó la biblioteca de secciones eficaces de fotones *eprdata14*.

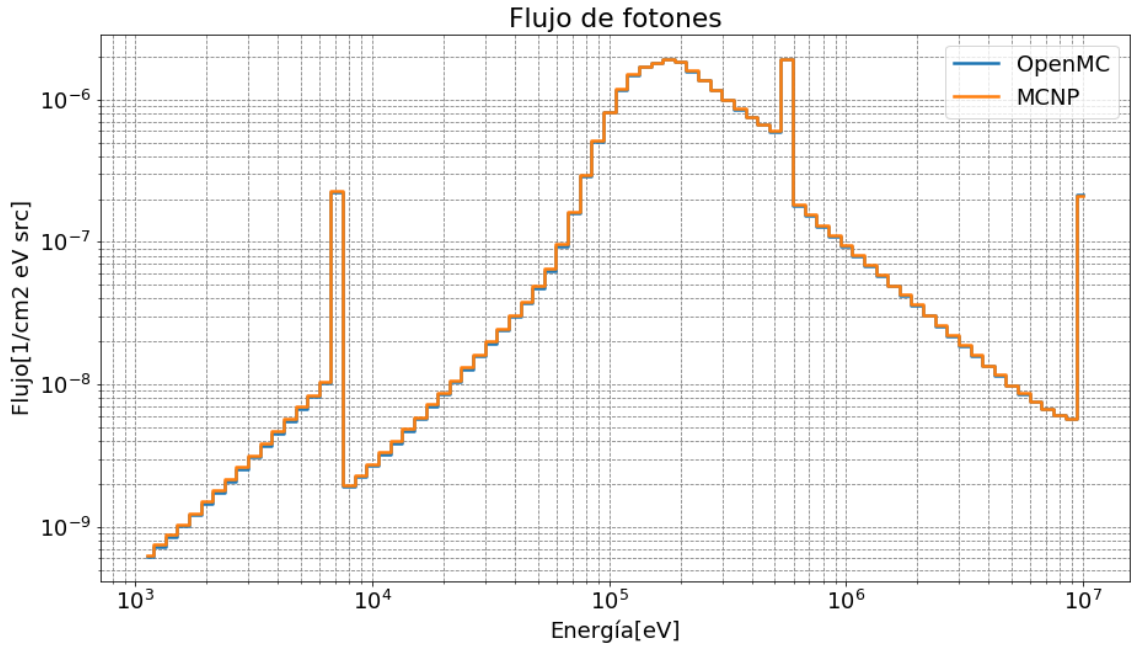

Figura 2.5: Espectro del flujo promedio en el volumen de la esfera utilizando los códigos de cálculo OpenMC y MCNP.

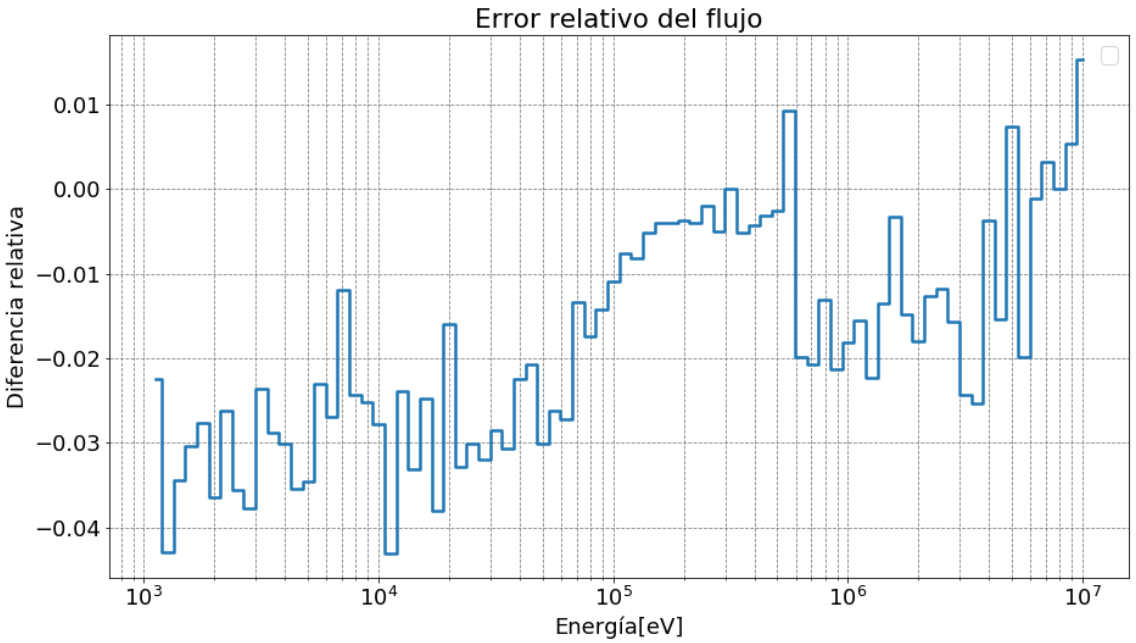

Figura 2.6: Diferencia relativa del espectro obtenido con OpenMC respecto del calculado con MCNP.

Los resultados presentados en el informe técnico del benchmark numérico fueron obtenidos utilizando la biblioteca de secciones eficaces de fotones *eprdata12*. De igual manera se observa que los resultados obtenidos se corresponden con los presentados en el informe técnico en su figura 4, ya que en ambos casos la diferencia porcentual del espectro obtenido con OpenMC y MCNP se encuentra dentro de una banda de

4 % de diferencia porcentual. De esta forma se verifica el correcto funcionamiento del transporte de fotones en OpenMC.

Capítulo 3

Métodos de reducción de varianza

3.1. Métodos de reducción de varianza

En la figura 3.1 se puede ver un haz de partículas incidiendo sobre un blindaje y un tally del otro lado del mismo, utilizado para estimar la dosis en dicha posición. Como el objetivo del blindaje es justamente hacer que muy pocas partículas que inciden sobre el mismo la puedan atravesar, se tiene que la relación entre partículas que llegan al tally respecto de las simuladas en total es muy baja. Es por ello por lo que, en estos casos se tienen varianzas muy grandes, debido a la poca cantidad de observaciones que se llevan a cabo en el tally. Por esto, son sistemas en los que es particularmente interesante la aplicación de métodos de reducción de varianza.

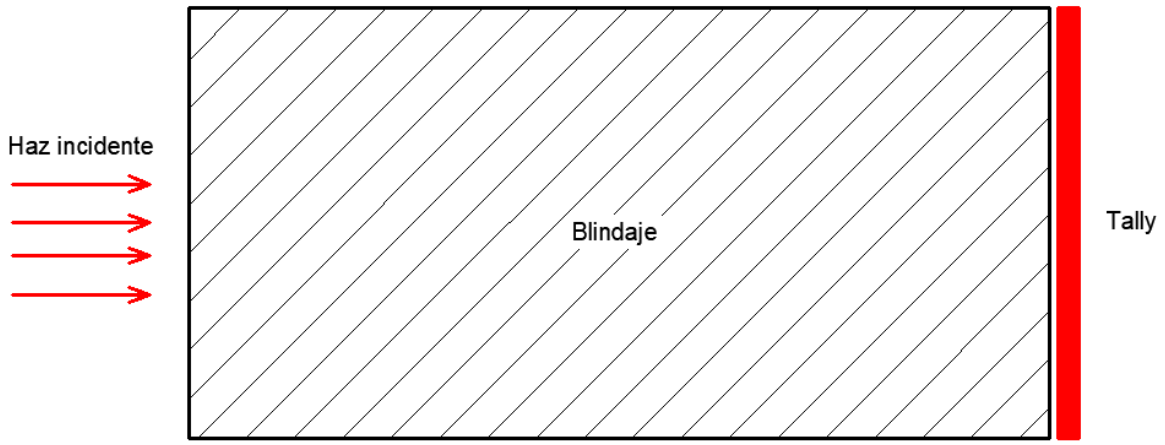

Figura 3.1: Esquema representativo de un haz de partículas incidentes sobre un blindaje, en el cual se coloca un tally en el extremo opuesto al haz.

Los métodos de reducción de varianza se encargan de obtener una precisión requerida en un menor tiempo computacional. Esta precisión depende en general del tipo de tally empleado y la forma en la cual se hace el muestreo del camino de la partícula. La idea es simular la mayor cantidad de partículas que sean importantes, es decir,

aquellas que tienen alta probabilidad de contribuir al tally a expensas de aquellas que difícilmente puedan afectar al mismo.

3.1.1. Figura de mérito *FOM*

Para poder cuantificar la eficiencia de la simulación, resulta de particular interés tener una figura de mérito que tenga en cuenta el tiempo utilizado en el transporte de las partículas y la varianza de las observaciones realizadas. Si la cantidad de partículas simuladas es N y el tiempo total de transporte de partículas es T se tiene que $N \propto T$, debido a que mayor cantidad de partículas implica mayor tiempo de simulación. Definiendo el error relativo R como el cociente entre la estimación de la desviación estandar de la media $\sigma_{\bar{x}}$ y la estimación de la media $\bar{x}$. Por otro lado, utilizando la ecuación 1.17 se tiene que $R \propto \frac{1}{\sqrt{N}}$. Por lo tanto definimos a la figura de merito como:

$$FOM = \frac{1}{R^2 T} \tag{3.1}$$

Una disminución en el tiempo de cálculo o en el error relativo conlleva a aumentar el *FOM*, que es justamente lo que se busca con los métodos de reducción de varianza. Es por ello por lo que dadas dos simulaciones Monte Carlo de transporte de partículas, con las mismas definiciones de geometría, materiales y fuente de partículas, aquella que tenga mayor *FOM* es la que resulta ser más eficiente. Las técnicas utilizadas para reducir la varianza del problema en general conllevan a un aumento de T pero una mayor disminución de R^2, aumentando en consecuencia la *FOM*.

3.2. Captura implícita o survival biasing

La captura implícita es un método sencillo de reducción de varianza en el cual la partícula no se puede absorber sino que simplemente se disminuye el peso de la misma en cada colisión. Este peso permite que en materiales altamente absorbentes, la partícula no se absorba y continúe su historia a expensas de la disminución del peso de la misma.

Esta técnica ya se encuentra implementada en OpenMC para transporte de neutrones en la version estable y se puede observar el manual de física de neutrones [20] para mayor información sobre su funcionamiento.

3.3. Geometry splitting y ruleta rusa

La utilización de *Geometry splitting* junto con la ruleta rusa es uno de los métodos de reducción de varianza más viejos y más utilizados en códigos Monte Carlo [3]. Es

por ello por lo cual se lo eligió para implementarlo en OpenMC.

La ruleta rusa consiste en que si una partícula de peso w tiene un peso inferior a un peso de corte w_c, esta tiene una probabilidad $P_{vivir} = \frac{w}{w_s}$ de continuar con un peso w_s y una probabilidad $P_{morir} = 1 - \frac{w}{w_s}$ de que se termine la historia de la partícula.

El método *Geometry splitting* consiste en asignar a cada celda dentro de la geometría una importancia dada, aumentando la misma en dirección al tally y disminuyendo en el caso contrario. La importancia no necesariamente tiene que ser un número entero sino que puede ser cualquier número real positivo. Cuando la partícula i con peso w_i cruza una superficie la cual separa a dos celdas con distintas importancias sufre un cambio en su peso dependiendo de la relación $C = \frac{I_{new}}{I_{prev}}$ entre la importancia de la celda previa I_{prev} y la importancia de la nueva celda I_{new}.

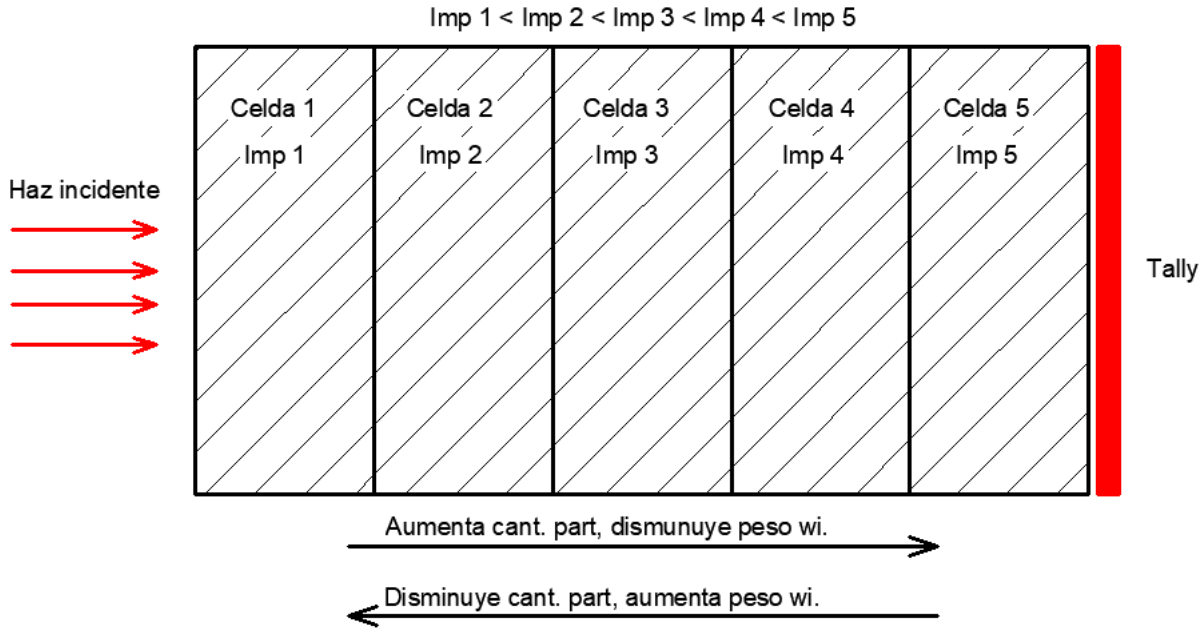

Figura 3.2: Esquema de Geometry splitting. Este método busca que la cantidad de partículas aumente en la dirección en la cual aumenta la importancia y que disminuya en el caso contrario.

Para el caso en el cual $C > 1$ se tiene que la partícula pasa a una celda de mayor importancia. Se define entonces a n como el mayor entero dentro del conjunto $[1, C]$ y a p como $p = C - n$. Luego, con una probabilidad p se generan $n + 1$ partículas y con probabilidad $1 - p$ se tienen n partículas. En cada caso el peso de cada nueva partícula será de $\frac{w_i}{C}$.

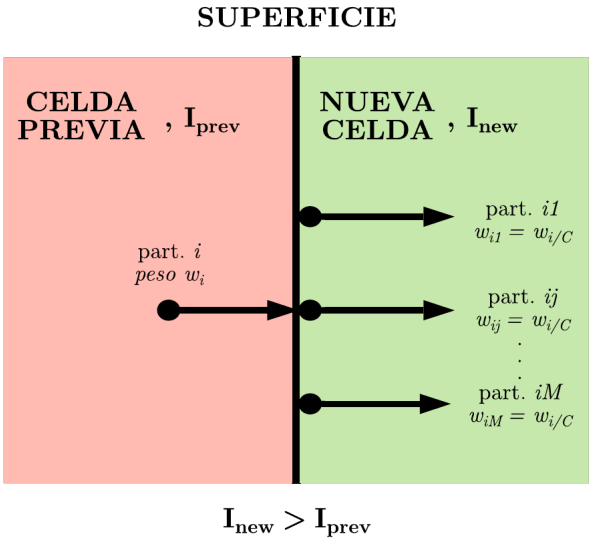

Figura 3.3: Esquema representativo de la división de una partícula al entrar a una celda de mayor importancia.

Como se vio en la sección de tallies 1.2, es importante conservar el peso de las partículas de manera que el estimador del valor medio $\hat{x}_w$ tenga esperanza $\bar{x}$ y el problema mantenga la física relacionada al mismo. Planteando la suma de los pesos de las partículas con sus probabilidades se demuestra a continuación, que para el caso de $C > 1$ el peso w_i de la partícula i es conservado.

$$
\begin{aligned}
&= p(n+1)\frac{w_i}{C} + (1-p)n\frac{w_i}{C} \\
&= pn\frac{w_i}{C} + p\frac{w_i}{C} + n\frac{w_i}{C} - pn\frac{w_i}{C} \\
&= (p+n)\frac{w_i}{C} \\
&= w_i
\end{aligned}
$$

Por otro lado, para el caso en el cual $C < 1$ se tiene que la partícula pasa a una celda de menor importancia. En este caso se aplica la ruleta rusa de manera que con probabilidad $1-C$ se termina la historia de la partícula, mientras que con probabilidad C la partícula sobrevive con peso $\frac{w_i}{C}$.

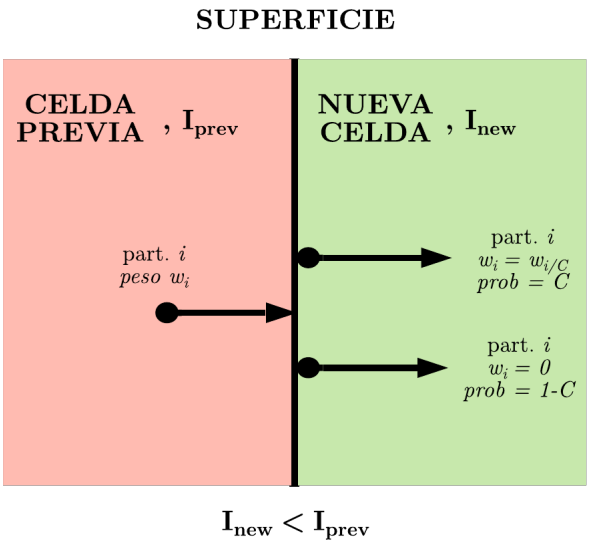

Figura 3.4: Esquema representativo de la división de una partícula al entrar a una celda de menor importancia.

Nuevamente planteando la suma de los pesos de las partículas con sus probabilidades se demuestra que para el caso de $C < 1$ el peso de la partícula i es conservado.

$$= (1 - C)0 + C\frac{w_i}{C}$$
$$= w_i$$

El peso de las partículas que sobreviven a la ruleta rusa es incrementado de forma aleatoria de manera que, teniendo en cuenta las partículas descartadas por bajo peso, el peso global de las mismas se conserve.

En el estado actual del código no se toma acción sobre partículas que ingresan a una zona de muy baja densidad, ya que la probabilidad de que la partícula no llegue a otra superficie de la celda es nula. Dividir partículas en vacío conlleva entonces a aumentar el tiempo de simulación de cada una y por lo tanto se evita que esto suceda en la implementación del código.

Por lo tanto, se puede decir que se conserva el peso inicial de la partícula en ambos casos, y en caso de que el tally se encuentre ubicado en una zona de mayor importancia que la celda inicial de la partícula se obtendrá una reducción de la varianza.

Este método funciona correctamente en casos donde no existe una dependencia angular muy alta, debido a que si muy pocas o ninguna partícula incide en una celda importante no se dividirán lo suficiente [3].

Información más detallada del funcionamiento del método puede ser obtenida en [3, 12, 21].

3.3.1. Implementación del código

Las modificaciones más importantes realizadas al código para implementar el método de reducción de varianza por *Geometry Splitting* fueron las siguientes:

1. Se añadió al archivo `settings.py` perteneciente al *API* de OpenMC la capacidad de definir si la simulación que se va a llevar a cabo tiene que utilizar la técnica de reducción de varianza *Geometry splitting*, de manera que al exportar el archivo `settings.xml` ésta información se encuentre incluida.

2. En el archivo `settings.cpp` del *código fuente* se generó la capacidad de determinar el nuevo modo de simulación implementado en el archivo `settings.xml`.

3. Se modificó el archivo `cell.py` perteneciente al *API* de OpenMC para incluir como una propiedad de cada celda a la importancia de la misma, de manera que al exportar el archivo `geometry.xml` esta información se encuentre incluida.

4. En el archivo `cell.cpp` del *código fuente* se generó la capacidad de leer la nueva propiedad de la celda dada en el archivo `geometry.xml`. Se implementó como propiedad de las celdas a la importancia.

5. En el archivo `particle.cpp` del *código fuente* se implementó como una propiedad de la clase `Particle()` la importancia de la celda en la que se encuentra y la importancia de la celda previa. Además se añadieron dos métodos a dicha clase las cuales cuentan con la lógica de esta técnica. Estos métodos son:

    - `void Particle::get_importance();`: Es la encargada de determinar la importancia que tiene la celda en la cual se encuentra la partícula.
    - `void Particle::geometry_splitting();`: Es la encargada de aplicar la técnica de reducción de varianza propiamente dicha.

El método `void Particle::geometry_splitting();` es ejecutado cada vez que la partícula atraviesa una superficie. Utilizando la importancia de la celda previa y la celda actual, las cuales son propiedades de cada partícula, se efectuá la lógica que fue explicada en esta sección.

Las importancias son asignadas por el método `void Particle::get_importance();` cada vez que una partícula atraviesa una superficie, debido a que hay que determinar la importancia de la celda a la cual se dirige la misma, es decir, determinar I_{new}. En dicho caso, la importancia I_{new} que tenia la partícula antes de este evento pasa a ser la importancia previa I_{prev}. Es por ello por lo cual también es necesario asignar la importancia de la celda de la cual una partícula proviene, para que al incidir en una superficie de celda, se cuente con la I_{prev}.

En la sección del apéndice B.1 se pueden observar los códigos de ambas funciones.

3.3.2. Ruleta rusa como control poblacional

Este método consiste en aplicar la ruleta rusa a cada partícula que alcance un peso menor o igual a un peso limite denominado weight cutoff. Esta técnica es utilizada principalmente en conjunto con los métodos de reducción de varianza *Survival biasing* y *Geometry splitting*.

El primero cuenta con la problemática de que en cada colisión la partícula pierde un porcentaje de su peso y puede conllevar a tener partículas de muy poco peso ya que la única forma de que termine la simulación de la misma es fugándose. Esto conlleva a simular partículas que al tener un peso tan bajo su aporte al tally es pequeño a comparación del tiempo de simulación empleado en la misma.

La misma problemática se presenta en *Geometry splitting*, ya que si bien se busca tener partículas de menor peso en dirección al tally, según cual sea el camino de la partícula, esta puede terminar con un peso muy pequeño.

El peso de las partículas que sobreviven a la ruleta rusa es incrementado de forma aleatoria de manera que, teniendo en cuenta las partículas descartadas por bajo peso, el peso global de las mismas se conserve.

Esta técnica ya se encuentra implementada en el código, pero fue pensada para ser usado solamente con captura implicita [20]. Esto tiene la problemática de que el peso de corte para el cual se aplica la ruleta es el mismo independientemente de la celda en la cual se encuentre la partícula. Pero como en *Geometry splitting* se busca que cada celda cuente con su importancia y por lo tanto se busca partículas de un peso determinado en cada una, es necesario cambiar el código de manera en que para celdas de mayor importancia el peso de corte sea más bajo que para celdas de menor importancia.

Esto se logró haciendo que el peso de corte w_c el cual es ingresado por el usuario sea $\frac{w_c}{imp_i}$, donde imp_i es la importancia de la celda i. Por otro lado, si la partícula sobrevive a la ruleta rusa se tiene que su peso será w_s el cual fue ingresado por el usuario independientemente de la celda en la cual se encuentre la partícula. Por lo tanto, este peso también fue modificado de manera que el peso resultante de la partícula será $\frac{w_s}{imp_i}$ debido a que se busca tener partículas de menor peso en celdas de mayor importancia, y que una partícula de alto peso afecta a la estadística del tally. Se define entonces la probabilidad de que la partícula sobreviva como $\frac{w_i imp_i}{w_s}$.

Siempre que se utilice *Survival biasing* o *Geometry splitting*, la ruleta rusa de control poblacional será aplicada con esta nueva implementación.

En el apéndice B.2 se puede observar el código implementado para este método.

3.4. Weight window

Esta técnica de reducción de varianza consiste en definir una ventana de pesos en cada celda. Cada vez que una partícula incide sobre dicha celda y este método se encuentre funcionando, se buscará que la misma tenga un peso que se encuentre dentro de dicha ventana. Esta es una de las técnicas de reducción de varianza que se implementó en el *código fuente*.

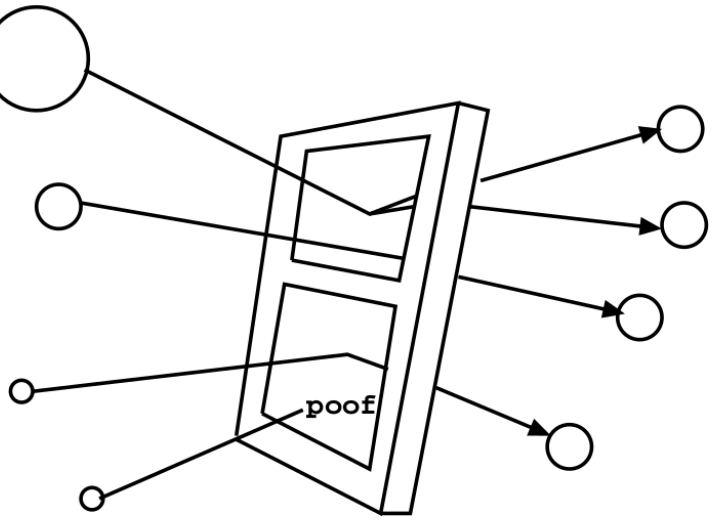

Figura 3.5: Esquema representativo del funcionamiento del método de reducción de varianza *Weight window*. Se puede ver que una partícula con peso superior al de la ventana se divide en otras con un peso tal que estas se encuentran dentro de la ventana. Si la partícula tiene un peso inferior al de la ventana esta puede terminar su historia (representada con "poof" en el gráfico) o terminar con un peso que se encuentre dentro de la ventana. Imagen tomada de [3].

Si la partícula tiene un peso w que se encuentra por debajo del peso inferior de la ventana w_{inf} se ejecuta la ruleta rusa para definir si esta sobrevive o no. En caso de que sobreviva se le asigna un peso w_{surv} tal que esta se encuentre dentro de la ventana.

Por otro lado, si la partícula tiene un peso w superior al peso superior de la ventana w_{sup} la misma es dividida en una cantidad de partículas tal que, conservando el peso w de la misma, cada una de ellas tenga un peso que se encuentre dentro de la ventana.

La definición del peso superior de la ventana w_{sup}, como así también el peso que tienen las partículas al sobrevivir la ruleta rusa w_{surv} son asignados mediante un factor constante, C_{sup} y C_{surv} respectivamente, respecto del peso inferior de la ventana w_{inf}. De esta manera solamente hay que definir el peso inferior de la ventana para cada celda, ya que las constantes recién mencionadas son las mismas para toda la geometría.

Al igual que en *Geometry splitting* se busca que el peso de las partículas disminuya en dirección a la zona importante o de interés, aumentando la cantidad de partículas simuladas en dicha dirección. Es por ello por lo cual se define al peso inferior de la ventana w_{inf} como la inversa de la importancia que tiene la celda, el cual es la misma

forma en la cual MCNP asigna dicho peso [3, 13].

La constante que define el limite superior de la ventana se lo define como $C_{sup} = 5$, ya que esta relacion entre los limites de ventanas fue probado que funciona "bien" [3]. Por otro lado, a la constante que define el peso de la partícula al sobrevivir a la ruleta rusa se eligió que sea $C_{surv} = 2{,}5$, entonces estas partículas tendrán un peso que se encuentra en el valor medio de la ventana.

Esta técnica puede ser utilizada cuando una partícula tiene una colisión y/o cuando la partícula cruza una superficie para dirigirse a otra celda. La utilización del método en cada colisión tiene sentido cuando la partícula cambia su peso a medida que tiene colisiones, como es el caso del survival biasing. Si un método como este no es utilizado solamente se debe utilizar sobre las superficies ya que no hay forma de que la partícula cambie su peso una vez que se encuentra dentro de una celda. Por otro lado, cuando una partícula cambia de celda, la ventana de peso a la que tiene que pertenecer cambia y por lo tanto en este caso si es necesario la aplicación del método.

A diferencia de los métodos *Geometry splitting* y *Survival biasing*, esta técnica no necesita un control de población mediante una ruleta rusa debido a que el limite inferior de la ventana ya cumple dicho rol.

Por otro lado, una gran diferencia entre el método *Geometry splitting* y *Weight window* es que, la primera funciona utilizando una relación entre importancias de dos celdas, mientras que la ultima se utiliza de forma absoluta sobre la importancia de cada celda. Además la técnica *Geometry splitting* es efectuada independientemente del peso que tiene la partícula, mientras que para *Weight window* la acción que se toma sobre la partícula depende de su peso.

3.4.1. Implementación del código

Las modificaciones más importantes realizadas al código para implementar el método de reducción de varianza *Weight window* fueron las siguientes:

1. Se añadió al archivo `settings.py` perteneciente al *API* de OpenMC la capacidad de definir si la simulación que se va a llevar a cabo tiene que utilizar la técnica de reducción de varianza *Weight window*, de manera que al exportar el archivo `settings.xml` esta información se encuentre incluida. Se puede elegir si el método es aplicado solamente al cruzar una superficie o al colisionar en la celda, o que se aplique en ambos casos.

2. En el archivo `settings.cpp` del *código fuente* se generó la capacidad de determinar el nuevo modo de simulación implementado en el archivo `settings.xml`.

3. Se modificó el archivo `cell.py` perteneciente al *API* de OpenMC para incluir como una propiedad de cada celda los 3 pesos necesarios para la utilización de este

método, de manera que al exportar el archivo `geometry.xml` esta información se encuentre incluida.

4. En el archivo `cell.cpp` del *código fuente* se generó la capacidad de leer las nuevas propiedades de la celda dada en el archivo `geometry.xml`. Se implementó como propiedades del celda a los 3 pesos w_{inf}, w_{sup} y w_{surv}.

5. En el archivo `particle.cpp` del *código fuente* se implementó como una propiedad a la clase `Particle()` los pesos de la celda en la que se encuentra. Además se añadieron dos métodos a dicha clase, los cuales cuentan con la lógica de esta técnica. Estos métodos son:

   - `void Particle::get_window();`: Es la encargada de determinar los pesos de la ventana que tiene la celda en la cual se encuentra la partícula.
   - `void Particle::weight_window();`: Es la encargada de aplicar la técnica de reducción de varianza propiamente dicha.

El método `void Particle::weight_window();` es ejecutado cuando la partícula tiene una colisión y/o cruza una superficie que separa a dos celdas de la geometría, según como haya sido definido por el usuario.

La probabilidad con la cual la partícula con peso w sobrevive a la ruleta rusa aplicada cuando la misma tiene un peso inferior al limite inferior de la ventana es $\frac{w}{w_{surv}}$.

Como se mencionó anteriormente, a cada partícula se le asigna como una propiedad los pesos de la ventana mediante el método `void Particle::get_window();`. Esto se lleva a cabo debido a que si *Weight window* se debe ejecutar en cada colisión que la partícula tiene dentro de una celda, se evita buscar que ventana hay que aplicar a la misma cada vez que esta tenga una colisión.

La asignación de los pesos de la ventana a la propiedad de la partícula se lleva a cabo siempre que una partícula ingrese a una nueva celda, como así también cuando la misma es nueva o recién generada.

En la sección del apéndice B.3 se pueden observar los códigos de ambas funciones.

Capítulo 4

Mapeo de importancias y generador de ventanas

4.1. Introducción

Como se mencionó en los capítulos anteriores en la descripción de los dos métodos de reducción de varianza implementados, la asignación de una importancia a cada celda conlleva a una mejora en la estadística que se tiene en el tally, lo cual es de gran importancia desde el punto de vista de la eficiencia computacional. La elección de estas importancias de forma arbitraria mediante la intuición, experiencia o mediante un método "prueba y error" puede ser dificultosa [3]. Es por ello por lo que se decidió implementar un código el cual lleve a cabo un mapeo de importancias de celdas y que a su vez genere los pesos para la ventana de las mismas. El objetivo es la utilización de este código para utilizarlo en conjunto con los métodos de reducción de varianza mencionados en los anteriores capítulos, los cuales son *Geometry splitting* y *Weight window*.

El proceso de asignación de importancias se realiza planteando que la celda de mayor importancia es aquella donde se cuenta con un tally o aquella zona donde se busca reducir la varianza del problema. El cálculo de las importancias de las demás celdas viene dado de forma relativa a esta última.

El código fue implementado en el módulo de funciones básicas del *API* de OpenMC de manera que es un preprocesamiento de los datos que el usuario debe llevar a cabo antes de correr el programa. Para ello se creó un archivo de *Python* denominado *weight_window_generator.py* el cual cuenta con dos *clases* de *Python*. En la primera se implementó un método el cual es capaz de generar una nube de puntos los cuales abarcan a toda la geometría que se quiere analizar y que luego van a ser analizados por la segunda clase que lleva a cabo el proceso de estimación de importancias propiamente dicho.

En el siguiente diagrama se puede observar el flujo del código utilizado para llevar a cabo el proceso de mapeo de importancias y generador de ventanas.

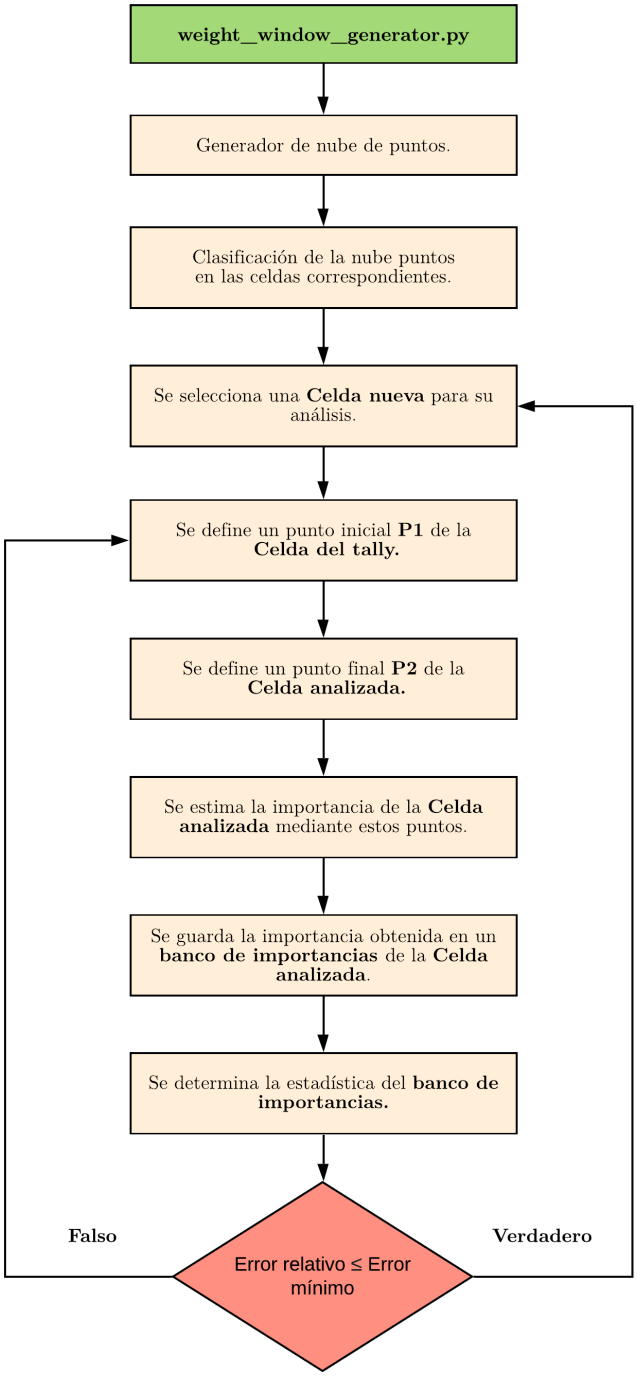

Figura 4.1: Diagrama de flujo del funcionamiento del generador de mapa de importancias.

4.2. Generador de puntos

Se genera una nube de puntos definiendo el extremo posterior superior derecho y el extremo anterior inferior izquierdo de un paralelepípedo, de manera tal que este encierre a la geometría que se quiere analizar. Luego, definiendo la cantidad de particiones que tiene cada dirección del paralelepípedo, se obtiene una nube de puntos. Para ello se utiliza la clase denominada *points_generator* la cual recibe por el usuario los siguiente inputs:

- Lista con las coordenadas del extremo posterior superior derecho.
- Lista con las coordenadas del extremo anterior inferior izquierdo.
- Discretización en la dirección x.
- Discretización en la dirección y.
- Discretización en la dirección z.

Se analiza como ejemplo el caso de una geometría sencilla como ser un cilindro y una esfera que se encuentran centrados en el origen de coordenadas como se puede ver en la siguiente figura:

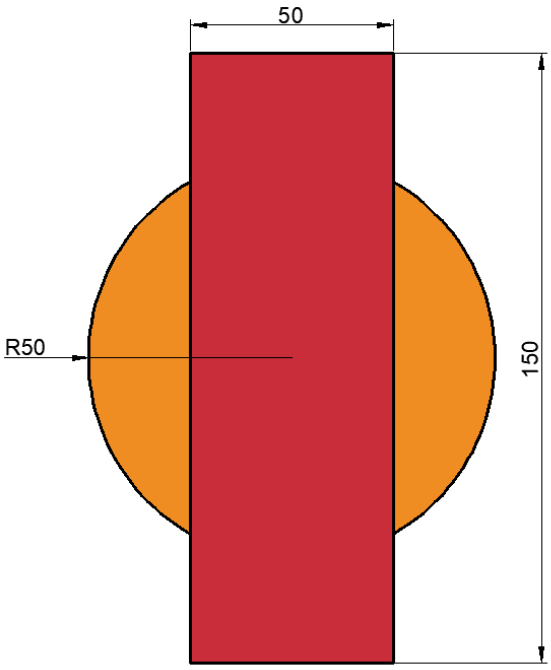

Figura 4.2: Corte axial de la geometría utilizada para pruebas dada por un cilindro y una esfera centrados en el origen de coordenadas. Las dimensiones están en cm.

Para este caso definiendo ambos extremos de la nube de puntos tal que encierre a la geometría recién descripta y con una discretización dada en cada dirección se obtiene una nube de puntos distribuida de forma uniforme en un paralelepípedo.

4.2.1. Clasificación de los puntos

La nube de puntos generadas anteriormente pasa a ser un input de la clase denominada *weight_ window_ generator* la cual es la encargada de analizar cada uno de los puntos para hacer el mapeo de importancias. El primer paso realizado es la clasificación de los puntos de la nube según la celda de la geometría a la cual pertenezca. En caso de que un punto no pertenezca a ninguna celda es descartado, ya que no aporta ninguna información para el mapeo de la importancia.

Esta clasificación se llevó a cabo utilizando una función denominada *find_ cell* del módulo *lib* del *API*, la cual identifica en que celda se ubica un punto correspondiente del espacio. Como se mencionó anteriormente, lo que tienen de particular las funciones del módulo *lib* es que estas ejecutan funciones del código fuente que son utilizadas para el transporte de partículas por OpenMC. Se optó por la utilización de esta función por dos motivos: el primero es debido a que ésta ejecuta una función del código fuente el cual al estar en *C++* tiene una mayor rapidez que una implementación en *Python* y el segundo es debido a que esta función está probada por los desarrolladores del programa de que funciona correctamente para muchas configuraciones geométricas que pueden ser difíciles de analizar como ser el caso de universos anidados.

De esta forma siguiendo el análisis llevado a cabo sobre la geometría definida en 4.2 se tiene que, separando a los puntos que se encuentran dentro de alguna celda y descartando los que no, se obtiene una nube de puntos con la forma de la geometría del problema como se puede observar en la figura 4.3.

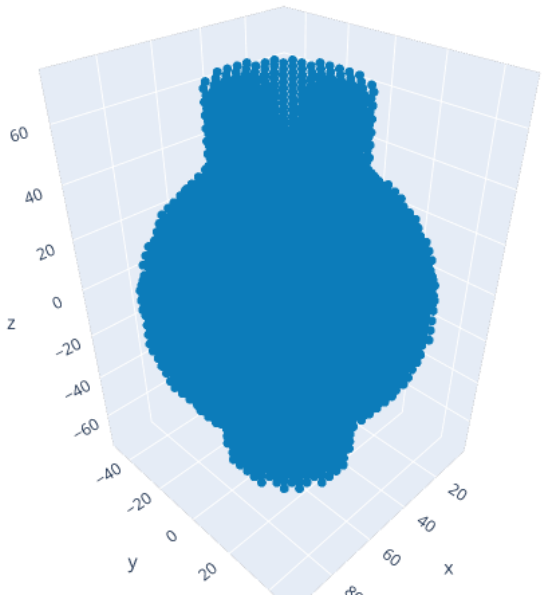

Figura 4.3: Nube de puntos con la definición de la geometría del problema.

En la figura 4.4 se pueden observar los puntos correspondientes a la esfera que rodea al cilindro central y en la figura 4.5 los puntos que se encuentran dentro del cilindro.

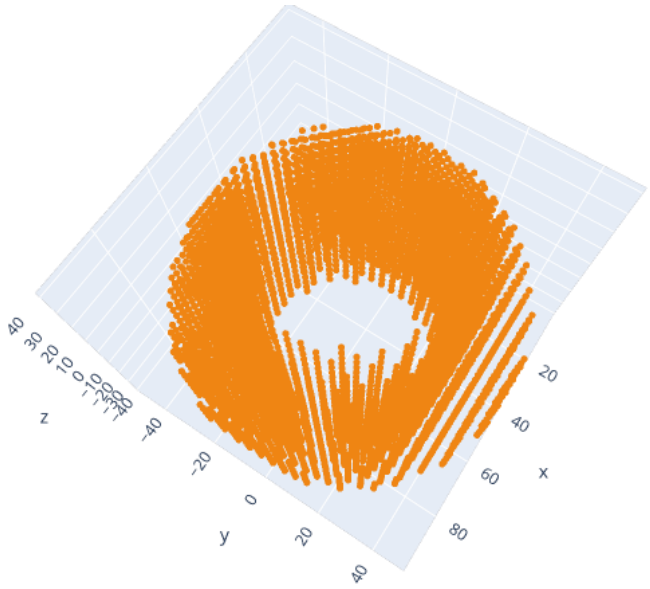

Figura 4.4: Nube de puntos correspondientes a la esfera que rodea al cilindro central.

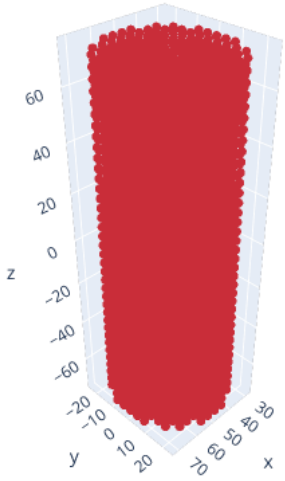

Figura 4.5: Nube de puntos correspondientes al cilindro central.

Como se puede ver la clasificación de los puntos en función de la celda a la que pertenecen se lleva a cabo de manera satisfactoria.

4.3. Estimación de importancia

Para llevar a cabo la estimacion de la importancia se utilizó la forma funcional de la componente espacial del esquema de importancias planteado en INIPOND [22], el cual es un modulo utilizado para estimar importancias en TRIPOLI [23].

Se realizaron simplificaciones sobre el mismo utilizando la sección eficaz total Σ_t en problemas de blindaje neutrónico y el coeficiente de atenuación lineal total en el caso de problemas de transporte de fotones, en lugar del coeficiente de Placzek. Por lo tanto se tiene que la variación de la importancia para estos casos está dada por la ecuación 4.1.

$$I(r) = I_0 \cdot exp\left(-\Sigma_t \cdot r\right) \tag{4.1}$$

Por otro lado, para medios difusivos se probó la utilización de la inversa de la longitud de difusión, obteniendo de esta manera una estimación de la importancia dada por la ecuación 4.2.

$$I(r) = I_0 \cdot exp\left(-\frac{r}{L}\right) \tag{4.2}$$

Para ambos casos r representa la distancia respecto del punto que se está analizando e I_0 es la importancia para $r = 0$.

4.3.1. Proceso de cálculo

Para llevar a cabo el mapeo de importancias se comienza eligiendo una celda en particular, se selecciona un punto $P2$ perteneciente a la nube de puntos de dicha celda y un punto $P1$ que se encuentra en la nube de puntos de la celda del tally. Dado estos dos puntos se define la dirección Ω_{12} la cual se corresponde a aquella que hay entre ambos puntos desde $P1$ hacia $P2$. En dicha dirección se determina la distancia d_i a la cual se encuentra la próxima superficie, que separa a esta celda de otra, partiendo desde $P1$.

Para la determinación de la distancia d_i se creó una nueva función en el módulo *lib* del *API* denominada *distance_to_boundary* la cual ejecuta la función del código fuente encargada de determinar distancias hasta la próxima superficie en el transporte de partículas. La función del código fuente ejecutada es la misma utilizada para llevar a cabo el transporte de las partículas por OpenMC.

Luego se lleva a cabo una traslación del punto $P1$ una distancia d_i en la dirección Ω_{12} previamente calculada. Este proceso se repite hasta que la distancia hasta la próxima superficie sea mayor a la distancia a la cual se encuentra el punto $P2$, es decir que $P1$ se encuentra en la misma celda que $P2$, caso en el cual d_i es igual a la distancia que separa a ambos puntos. En la figura 4.6 se puede observar un esquema del proceso recién mencionado.

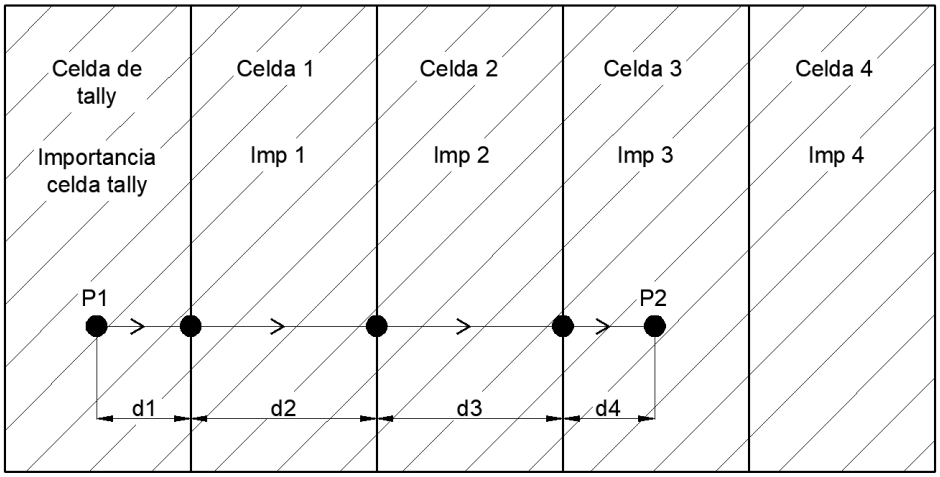

Figura 4.6: Esquema representativo del proceso llevado a cabo para determinar le importancia del punto $P2$ respecto del punto $P1$.

Como cada celda puede tener una definición de material distinta se debe realizar una estimación con la probabilidad de colisión de primer vuelo para cada una. Por este motivo se tiene una exponencial decreciente de la importancia diferente en cada celda, los cuales se pueden observar en la figura 4.7:

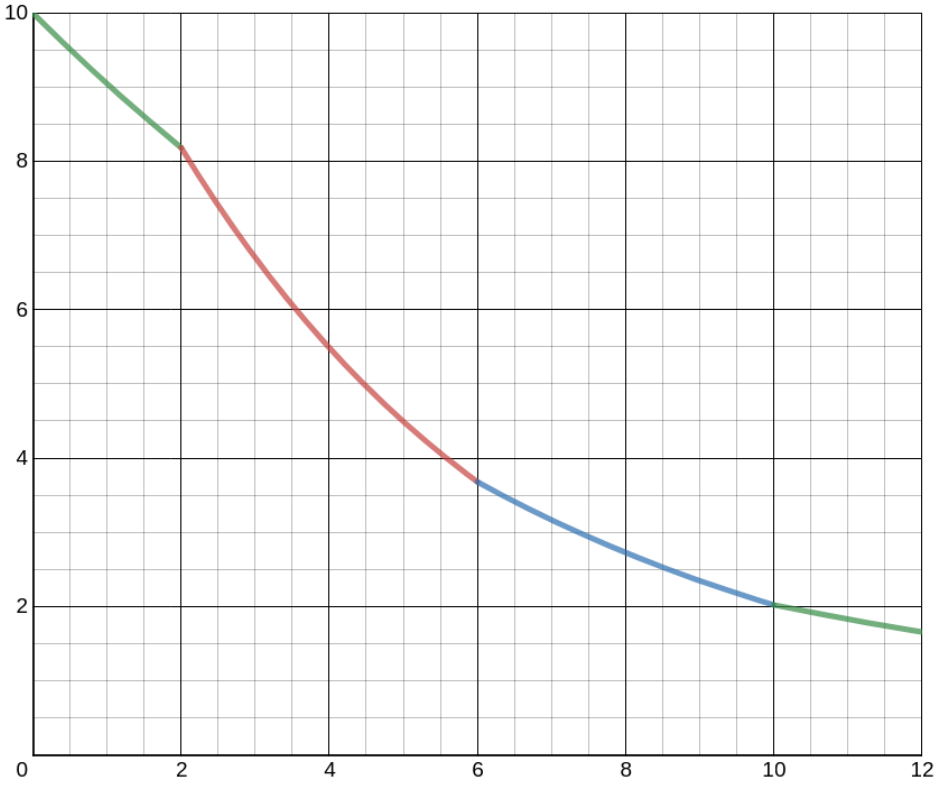

Figura 4.7: Evolución de la importancia con la distancia recorrida.

Se puede ver que el valor inicial I_0 de cada exponencial depende del punto final de la exponencial anterior. Entonces se puede escribir a la importancia del punto analizado $P2$ como una productoria de las probabilidades de colisión que tiene dentro de cada

celda como:

$$I_{P2} = I_{tally} \prod_i p(d_i) \tag{4.3}$$

donde nuevamente d_i es la distancia hasta la próxima superficie o hasta el punto $P2$ según corresponda, $p(d_i)$ es la función de opacidad de primer vuelo para una distancia d_i recorrida en la celda i e I_{tally} es la importancia de la celda del tally la cual fue ingresada por el usuario como input del código.

En caso de que la partícula recorra una distancia d_i en una celda en la cual hay vacío, se tiene que $p(d_i) = 1$ debido a que $\Sigma_t = 0$ y también que $L \longrightarrow \infty$, y por lo tanto la importancia no sufre una atenuación en el mismo.

Se busca que en todo el proceso de determinación de la importancia de una celda en particular no se analice la importancia entre los mismos puntos nuevamente para que el resultado obtenido sea lo menos sesgado posible. Para ello supongamos que la celda a analizar y la celda del tally tienen N y n puntos respectivamente, y que además $N > n$. En la figura 4.8 se puede ver un esquema de como se seleccionan los puntos finales e iniciales, de manera que la iteración sobre una variable i de una combinación de estos puntos que no se repita.

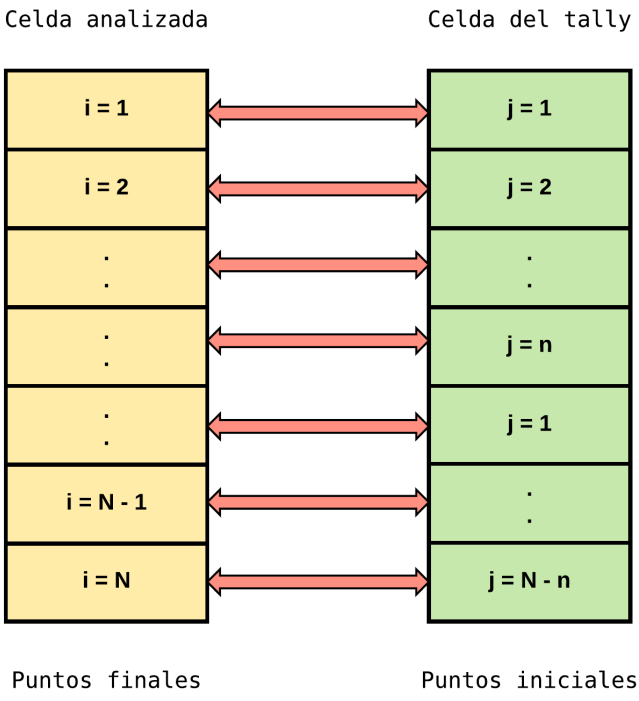

Figura 4.8: Combinación de puntos iniciales. El código itera sobre una variable i la cual va desde $i = 1$ hasta $i = N$. Para cada i se cuenta con una combinación de punto inicial y punto final las cuales definen una importancia. Para el caso analizado en donde $N > n$, se tiene que cuando $i = n + 1$ se vuelve a tomar el punto $j = 1$ de la celda del tally.

Dado que cada par de punto inicial y final analizado define una importancia, la importancia final de la celda será el valor medio de la importancia calculada para cada par de puntos. El proceso de cálculo de importancia en una celda finaliza cuando el error relativo de la importancia de la celda es menor o igual al especificado por el usuario como input del código.

Es por ello por lo que se decidió calcular la importancia total de la celda y su error relativo cuando la variable i sobre la cual se itera sea un múltiplo del 5 % de N. De esta forma se logra realizar un análisis del error relativo en el proceso de cálculo, y finalizar el mismo cuando se alcance la incerteza máxima buscada. Una vez que se hayan analizado la totalidad de combinaciones posibles, es decir N combinación de puntos, y todavía no se alcance el error relativo buscado lo que se hace es reordenar de forma aleatoria a la lista de los puntos que se tiene en ambas celdas. De esta manera se continua con el proceso con una nueva combinación de puntos finales e iniciales hasta alcanzar el error relativo deseado.

Este proceso se lleva a cabo para cada celda por separado obteniendo así el mapa de importancias de la geometría planteada.

4.3.2. Secciones eficaces

Para calcular la variación de la importancia utilizando las ecuaciones 4.1 y 4.2 se necesita llevar a cabo una estimación de la longitud de difusión L y la sección eficaz total Σ_t de los materiales de cada una de las celdas. El generador de mapa de importancias utiliza secciones eficaces condensadas a un grupo de energía.

Para ello se utiliza el módulo del *API* de OpenMC encargado de generar secciones eficaces multigrupo, con el cual se calculan las secciones eficaces condensadas a un grupo. La secciones eficaces están condensadas a un grupo utilizando el flujo producto de la configuración geométrica, de los materiales y la fuente del problema. La sección eficaz de absorción Σ_a, la sección eficaz total Σ_t y la sección eficaz de transporte Σ_{tr} a utilizar son obtenidas de esta forma. Este proceso genera una clase de *Python* que consiste en una librería con los resultados obtenidos y es la cual debe incluirse como input al código que lleva a cabo el mapeo de importancias.

Por definición se tiene que la longitud de difusión L es igual a:

$$L^2 = \frac{D}{\Sigma_a} \tag{4.4}$$

Por otro lado, el coeficiente de difusión D está definido como [24]:

$$D = \frac{1}{3\Sigma_{tr}} \tag{4.5}$$

Utilizando la definición del coeficiente de difusión se tiene entonces que la longitud

de difusión es:

$$L = \sqrt{\frac{1}{3\Sigma_a \Sigma_{tr}}} \tag{4.6}$$

4.3.3. Escaleo de la importancia

Al finalizar el proceso de mapeo de importancias es posible que en la posición en donde se tenga definida a una fuente puntual (fuente de partículas de peso unitario), la importancia de la celda sea distinta de 1. En caso que se utilice el método de reducción de varianza *Geometry splitting* esto no representa un mayor inconveniente debido a que este método utiliza una relación entre la importancia de las celdas y no así un valor absoluto.

Por otro lado, para el caso del método *Weight window* se tiene que cada vez que se ejecuta esta técnica se lo hace de forma absoluta sobre los pesos de la ventana. En este caso se busca que todas la partículas que nacen de una fuente se encuentren dentro de la ventana, ya que si no se debe analizar desde un principio a cada una de las partículas de fuente aumentando el tiempo que lleva la aplicación del método. Es por ello por lo que se realiza un escaleo a todo el mapa de importancias obtenido de manera que la celda donde se encuentra la fuente tenga importancia unitaria y por lo tanto las nuevas partículas se encuentren dentro de la ventana.

4.4. Generador de ventanas

Las ventanas utilizadas por el método de reducción de varianza denominado *Weight window*, son asignadas como fue mencionado anteriormente en la seccion 3.4. Por lo tanto se tiene que las ventanas de pesos asignado a la celda i es:

- Peso inferior: $w^i_{inf} = \frac{1}{I_i}$
- Peso superior: $w^i_{sup} = 5 \cdot w^i_{inf}$
- Peso al sobrevivir a la ruleta: $w^i_{surv} = 2{,}5 \cdot w^i_{inf}$

4.5. Análisis del tiempo del mapeo de importancias

Para llevar a cabo un análisis del tiempo de ejecución del código se procedió a analizar una geometría sencilla. Esta geometría consiste en 10 celdas concéntricas, en donde el material en todas ellas es agua. Se colocó una fuente puntual isotrópica de neutrones de 1MeV en el origen de coordenadas y se definió como la celda más

importante al casquete esférico exterior denominada celda 10. En la siguiente figura se puede ver la geometría definida:

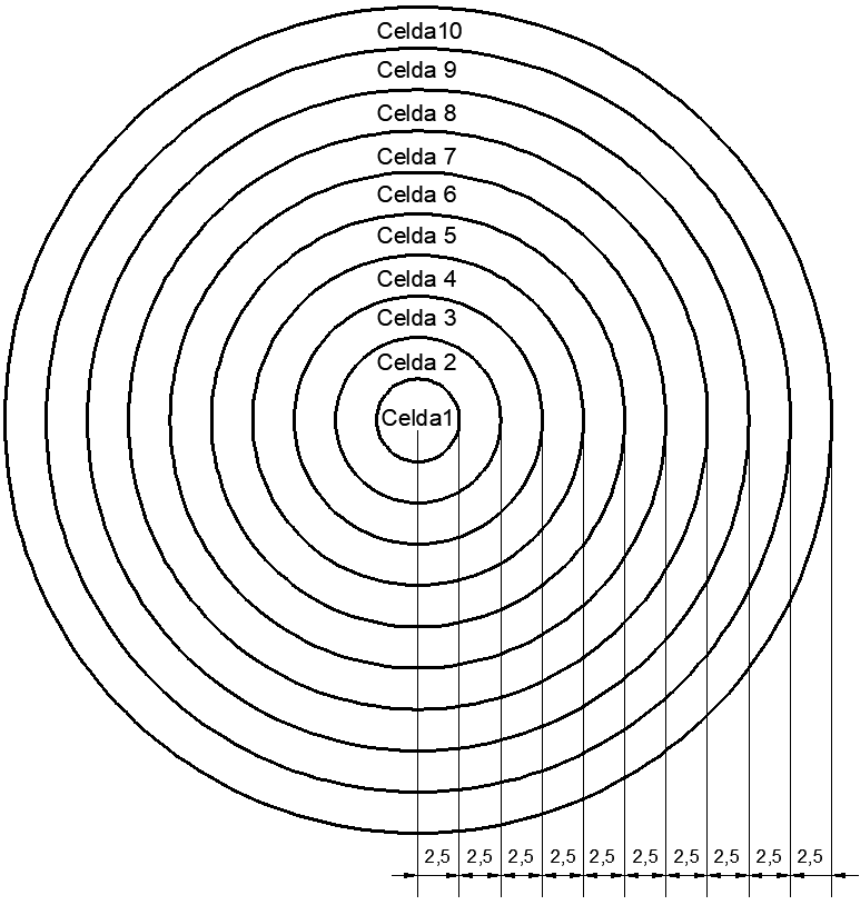

Figura 4.9: Geometría utilizada para llevar a cabo el análisis del tiempo de cálculo del mapa de importancias.

Se llevo cabo el análisis del tiempo de ejecución del código buscando un error relativo máximo de 1 % y se varió la cantidad de partículas totales a analizar. En la figura 4.10 se pueden observar los tiempos de ejecución total del código para las distintas cantidades de partículas.

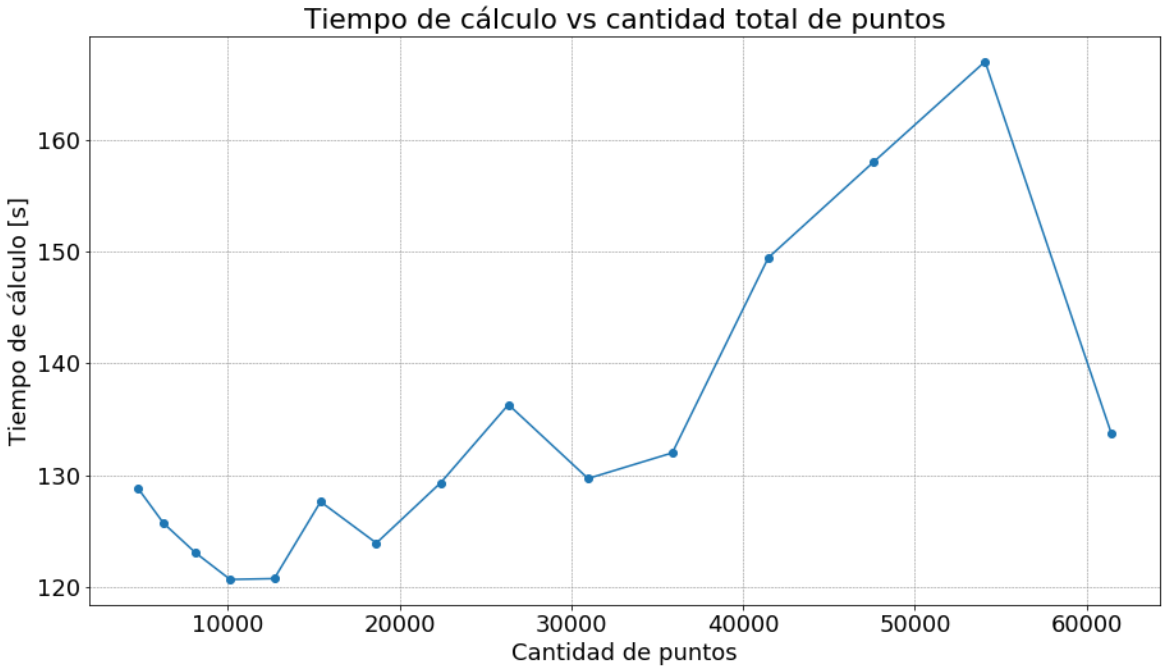

Figura 4.10: Tiempo de ejecución del generador de mapa de importancias en función de la cantidad de partículas iniciales de la nube de puntos.

Se puede observar que si bien existen variaciones en los tiempos de cálculo al aumentar el número de partículas, el órden es prácticamente el mismo y a comparación del tiempo de cálculo total que puede llegar obtenerse en una corrida de transporte Monte Carlo de gran cantidad de partículas, el mismo es despreciable. Por otro lado, el tiempo que tarda el código en calcular el mapa de importancia se debe comparar con el tiempo en el cual el usuario estima este mapa en base a su intuición y experiencia, ya que esta implementación busca reemplazar este trabajo.

En la figura 4.11 se muestran las importancias calculadas en cada celda, para las distintas cantidades de partículas iniciales de la nube de puntos, en función del tiempo total de cálculo empleado por el generador para esta geometría.

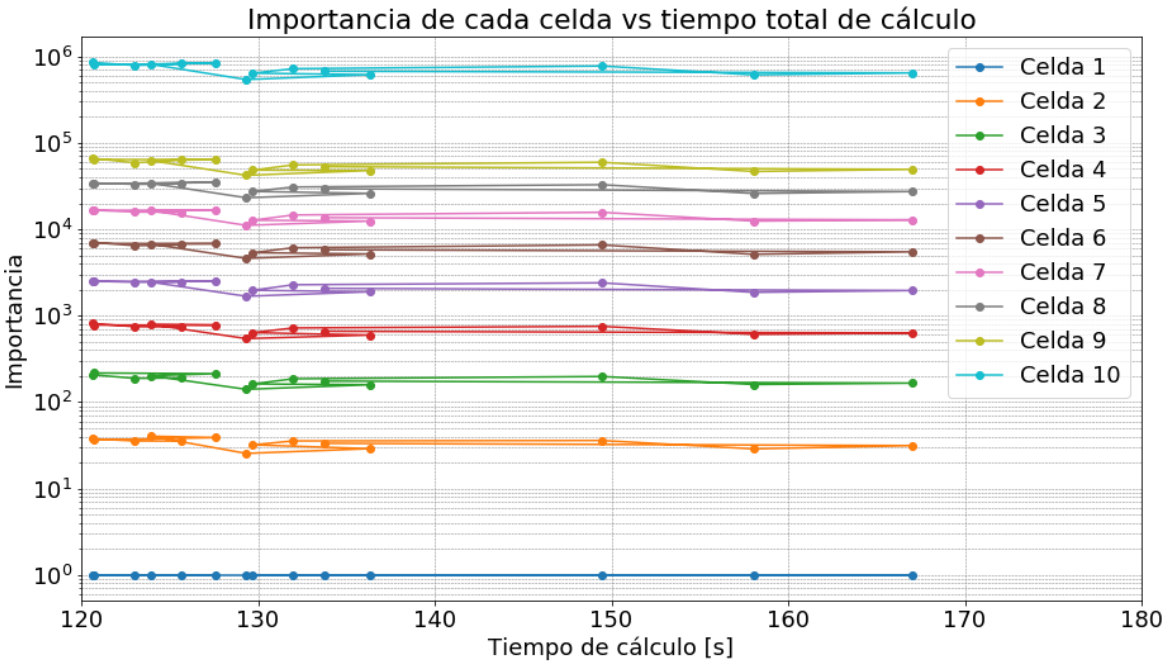

Figura 4.11: Importancia obtenida para cada celda en función del tiempo total de cálculo.

Se puede ver que el valor de la importancia de cada celda prácticamente no varía con el tiempo de cálculo empleado. Por último, se puede ver que como la celda 10 es la celda que simula el volumen en el que se encuentra el tally es aquella que tiene la mayor importancia.

La variación de la importancia entre celdas adyacentes difiere de algunas recomendaciones existentes [3], ya que es posible que al existir saltos de importancias grandes entre celdas la ejecución de la ruleta rusa se efectué muy frecuentemente. Por otro lado, como el peso de corte para el cual se ejecuta la ruleta rusa en ambos métodos de reducción de varianza implementados son inversamente proporcionales a la importancia de la celda, el peso de corte también disminuye con el aumento de la importancia.

4.6. Input del generador de mapa de importancias y de ventanas

Como se mencionó anteriormente el código requiere para su correcto funcionamiento una serie de datos los cuales son utilizados para generar el mapa de importancias. A continuación se mencionan los distintos inputs necesarios y la utilización de esa información en el código.

- Clase de Python del tipo *geometry*: Es utilizado para acceder a toda la información de la geometría del problema a analizar.

- Nube de puntos: Es el output de la nube de puntos generado por el "Generador de puntos". Es utilizado para iterar sobre los mismos, analizar a que celda pertenecen y llevar a cabo una estimación de la importancia.
- Error relativo: Indica que error relativo máximo se busca y es lo que define el limite del proceso iterativo de determinación de importancia de cada celda.
- ID de la celda de mayor importancia: En OpenMC cada celda tiene asignado un número entero que lo identifica. Esta información permite conocer a que celda asignar el valor de importancia de la celda más importante.
- Importancia de la celda del tally.
- Posición de la fuente: utilizado para llevar a cabo el escaleo del mapa de importancias.
- Librería de secciones eficaces: Es utilizado para determinar la probabilidad de no colisionar de las partículas.
- Tipo de estimación de importancias: Define con que método se estima la probabilidad de no colisionar de las partículas, es decir si se utiliza la ecuación 4.1 o 4.2. En caso de que la simulación sea de fotones, el código utilizara la estimación que utiliza Σ_t.

Capítulo 5

Aplicación de los métodos de reducción de varianza

5.1. Problema ejemplo para neutrones

Para llevar a cabo la validación de transporte neutrónico de las implementaciones realizadas sobre el código fuente, es decir los métodos de reducción de varianza y sobre el *API*, correspondiente al generador de mapa de importancias, se va a estudiar un problema sencillo para observar el comportamiento de la varianza ante la utilización de dichas implementaciones.

5.1.1. Geometría

La geometría del problema analizado consiste en un cilindro de 1m de alto y 50cm de radio de agua, el cual en el origen de coordenadas cuenta con una fuente puntual de neutrones de 1MeV. Por otro lado, se utilizaron 2 tallies de flujo neutrónico promedio en el volumen, ubicados a 25cm de la fuente cada uno. Cada volumen de los tallies consiste de una geometría de 5cm de radio y 1m de alto, esta última igual que el cilindro externo.

Luego se añadieron 14 celdas donde cada una de ellas es nuevamente un cilindro las cuales están centradas en el tally de volumen correspondiente a la celda 2. Cada una de estas celdas cuentan con un radio que va en aumento de a 5cm para lograr una mayor discretización del mapa de importancias en todo el volumen. La razón principal por la cual se optó por este tipo de geometría es porque se busca que el tally correspondiente a la celda 2, sobre la cual se centran las demás, sea más importante que el tally de la celda 3 y de esta manera poder analizar como cambia la incerteza de los espectros en ambos casos. En la siguiente figura se puede observar la geometría recién descrita:

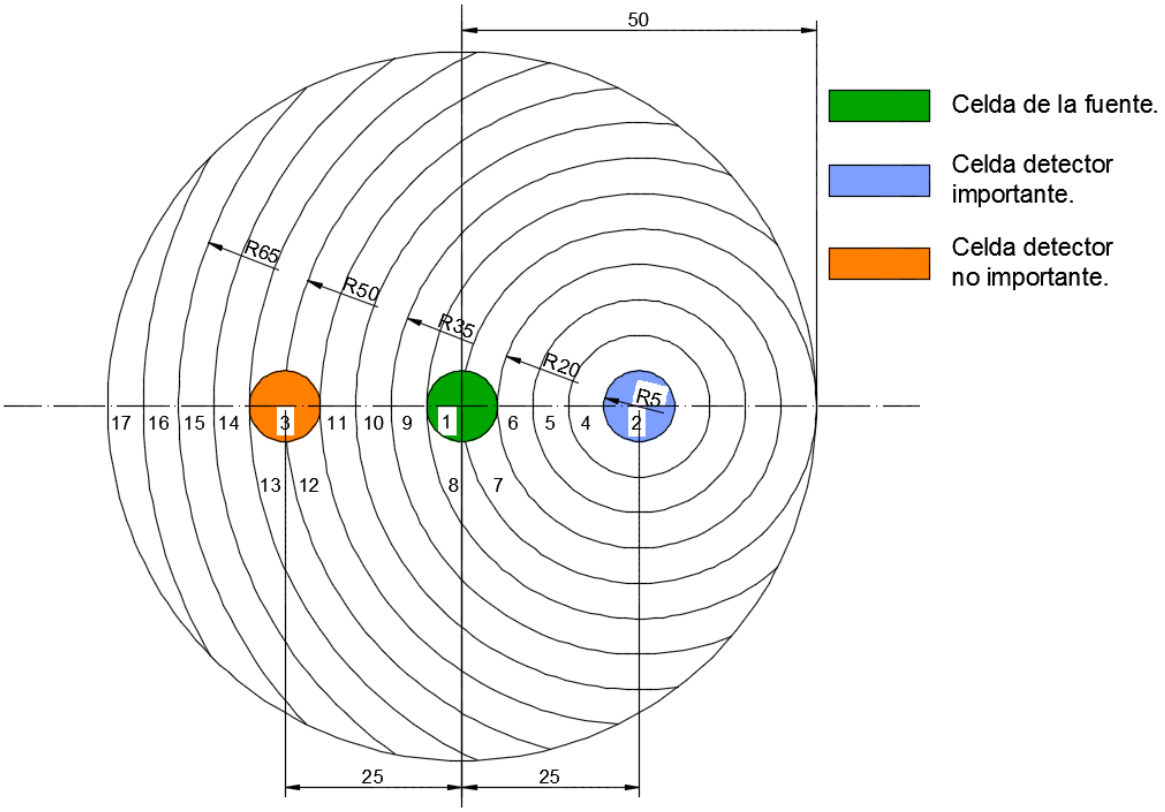

Figura 5.1: Geometría utilizada en el benchmark neutrónico. Cada celda tiene un forma cilíndrica y se pueden observar 3 celdas que resultan de mayor interés, las cuales se corresponden a la celda de la fuente puntual la cual se encuentra en el origen de coordenadas y las otras dos son tallies volumétrico. La unidad de todas las cotas es cm.

La geometría se planteó de forma sencilla para probar el funcionamiento de los métodos de reducción de varianza implementados, como así también del generador de mapa de importancias. Esta cuenta con el problema de tener celdas (como la celda 1 y 9) que comparten puntos de frontera y tendrán una importancia relativa entre ellas mayor a la recomendada.

5.1.2. Mapeo de las importancias

Mediante el uso del generador de mapa de importancias implementado en el *API* de OpenMC se llevó a cabo una estimación las importancias de cada celda. Para ello, lo primero que se realizó fue un cálculo de las secciones eficaces a 1 grupo de energía el cual es un input necesario para el generador de mapa de importancias. El grupo de energías utilizado fue de 0 a 1eV debido a que se espera un espectro mayoritariamente térmico en la zona del tally. Para condensar las secciones eficaces el *API* utiliza el espectro de neutrones correspondiente a la geometría y fuente recién señalada.

Luego, asignando como celda importante a la celda 2 y buscando un error relativo máximo en la importancia de cada celda de 1 % se obtuvieron las siguientes importan-

cias:

Celda #	Importancia	Celda #	Importancia	Celda #	Importancia	Celda #	Importancia
1	1,0	6	12,36	10	1,12	14	0,14
2	60,93	7	7,1	11	0,75	15	0,08
3	0,31	8	4,15	12	0,37	16	0,04
4	33,35	9	2,2	13	0,27	17	0,009
5	20,18						

Tabla 5.1: Mapa de importancias generado mediante el código implementado en el *API* de OpenMC.

Se puede observar de forma general que cuanto más cerca se encuentren las celdas respecto de la celda 2 (celda más importante), mayor importancia tendrán las mismas.

Una vez calculada la importancia de cada celda, el generador de mapa de importancias implementado asigna estas propiedades a los objetos openmc.Cell() de la geometría para luego ser exportados al archivo `geometry.xml`, como así también los pesos de las ventanas para la utilización del método *Weight window*.

5.1.3. Resultados

Para llevar a cabo un análisis de los métodos de reducción de varianza implementados en el código fuente se realizaron simulaciones utilizando los siguientes métodos:

- Sin método de reducción de varianza.
- *Geometry splitting*.
- *Weigh window* aplicado solamente cuando las partículas cruzan una superficie. Se lo denomina "Weight window (sup)".
- *Weigh window* aplicado cuando las partículas cruzan una superficie como así también cada vez que las mismas tienen una colisión. Además se utilizó el método survival biasing. Se lo denomina "Weight window (sup+col) + Surv. Bia.".

Utilizando el mapa de importancias calculado y con 1×10^7 cantidad de partículas de fuente, con una discretización de 90 bins de energía por década, se analizaron los resultados obtenidos en ambos tallies volumétricos. En la figura 5.2 y 5.3 se pueden observar los espectros de flujo de neutrones para el detector no importante y el error relativo de los mismos respectivamente, como así también para el caso del tally del detector importante sin métodos de reducción de varianza.

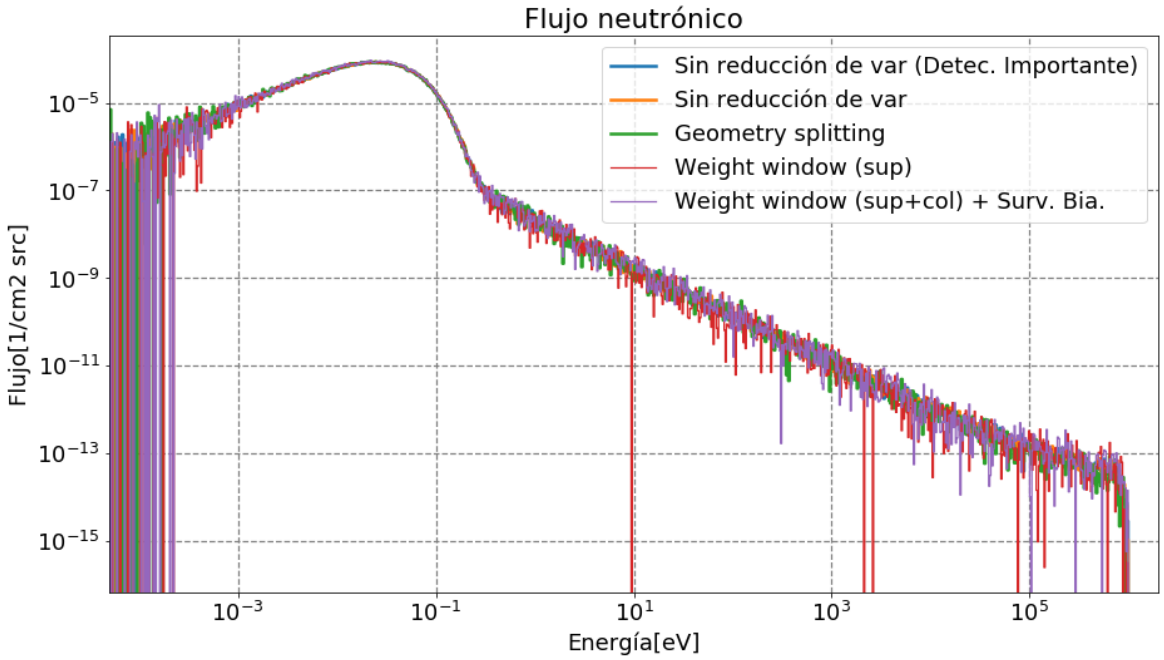

Figura 5.2: Espectros de flujo de neutrones utilizando los distintos métodos de reducción de varianza para el caso del tally de la celda no importante. También se presentan los espectros de flujo de neutrones obtenidos sin utilizar técnicas de reducción de varianza para ambos tallies.

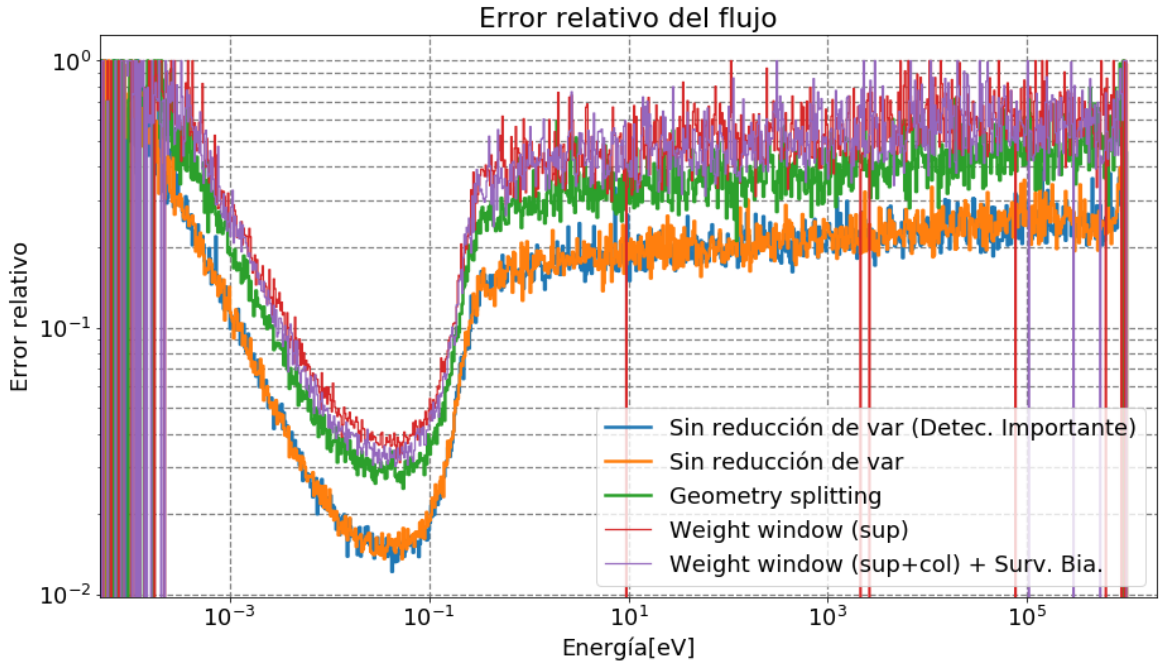

Figura 5.3: Error relativo de los espectros de flujo de neutrones presentados en la figura 5.2.

Se puede ver que independientemente del modo en el cual se lleva a cabo la simulación el espectro permanece constante. Además, se verifica que como la ubicación de los tallies es simétrica en la geometría, el espectro obtenido en el tally importante es el mismo que para el tally no importante.

Por otro lado, se puede observar que el error relativo es mayor cuando se aplican

las distintas técnicas de reducción de varianza respecto del caso en el que no. Esto se debe a que los neutrones que inciden sobre el tally de la celda no importante tienen peso mayor a la unidad y por lo tanto la varianza aumenta.

Es por ello por lo que se procedió a analizar los resultados producto del tally importante, es decir el de la celda 2, y se obtuvo el siguiente espectro de flujo neutrónico:

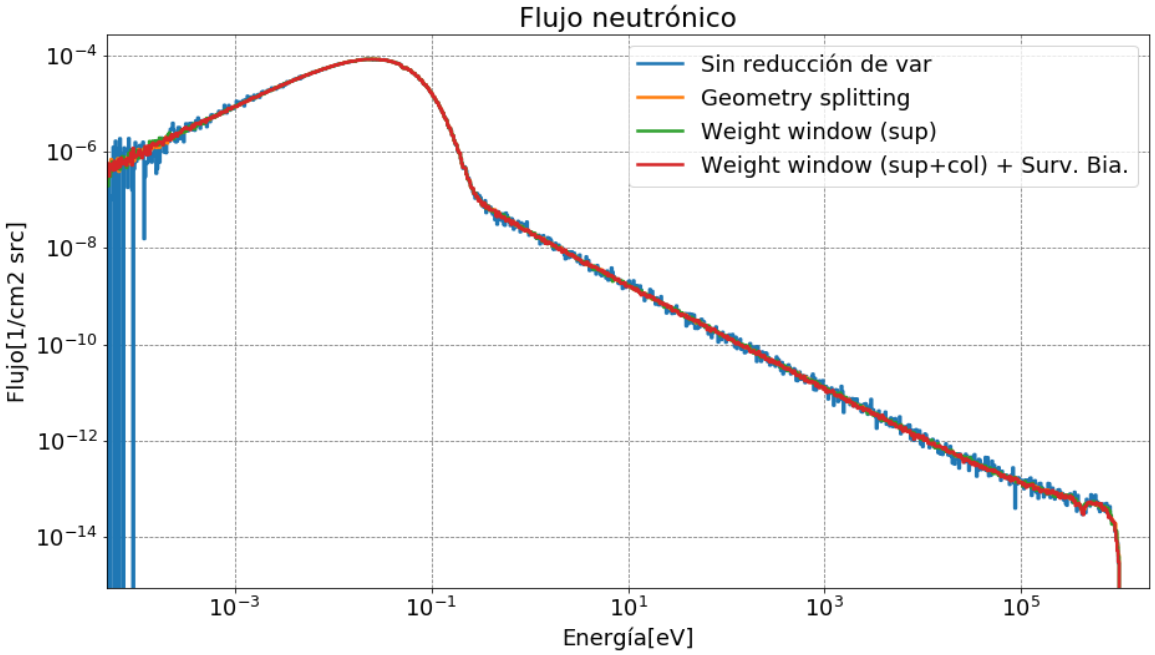

Figura 5.4: Espectro de neutrones resultante en el tally de flujo neutrónico en el volumen de la celda 2 para los distintos modos de simulación utilizados.

Se puede observar que el espectro del flujo es el mismo independientemente del método de reducción de varianza utilizado, lo cual concuerda con la conservación del peso de las partículas en la cual se basan estos métodos. Debido a que el único material dentro de toda la geometría es agua se obtiene un espectro térmico como era de esperarse debido al poder de moderación que tiene el mismo.

Los errores relativos obtenidos para cada espectro se encuentran en la figura 5.5.

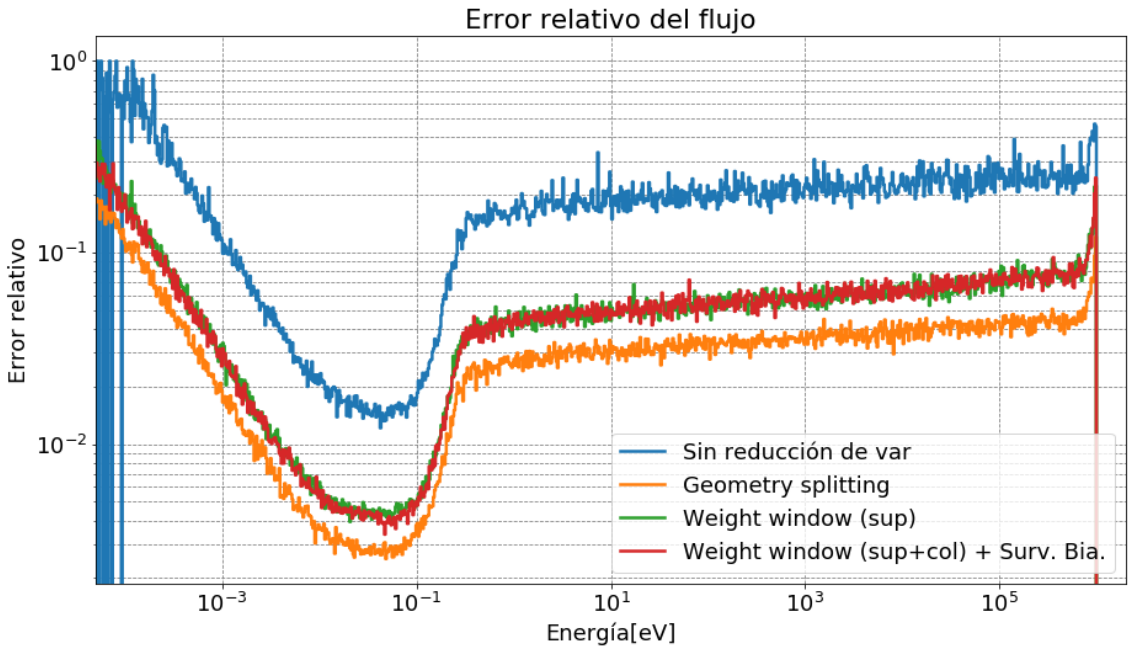

Figura 5.5: Error relativo del espectro de neutrones en el tally de flujo neutrónico de la celda importante para los distintos métodos de reducción de varianza utilizados.

Se verifica que para todo el rango de energías analizado la varianza disminuye respecto del caso en el cual no se utilizaron métodos de reducción de varianza. Esto concuerda con un correcto funcionamiento de las técnicas de reducción de varianza implementadas como así también del generador de mapa de importancias. Además se observa que el método de reducción de varianza *Geometry splitting* presenta una mayor disminución del error relativo.

El análisis de los espectros de los errores relativos obtenidos para cada método de reducción de varianza utilizado no alcanza para concluir que se tiene el resultado esperado. Esto se debe a que se busca obtener el menor error relativo en el menor tiempo posible, lo cual no se puede analizar en dicho gráfico. Es por ello por lo que se analizó la variación de la figura de mérito o *FOM* para distintas cantidades de partículas de fuente para cada método utilizado.

Para llevar a cabo este análisis, se determinó el error relativo que tiene el espectro para la energía en la cual el mismo cuenta con el pico del espectro, correspondiente a una energía de aproximadamente 10 meV. Se tomó el error en dicha energía ya que esta se corresponde con la menor varianza del error relativo y por lo tanto la incerteza de la misma es menor que en el resto del espectro.

En la figura 5.6 se graficó la variación del error relativo en función del tiempo de cálculo, mientras que en la figura 5.7 se graficó la variación de la *FOM* en función de la cantidad de partículas de fuente simuladas. En ambos casos se tuvo en cuenta el tiempo de cálculo empleado por el generador de mapa de importancias el cual fue de 50 segundos para este problema sumado a los 200 segundos utilizados para la generación

de las secciones eficaces a un grupo.

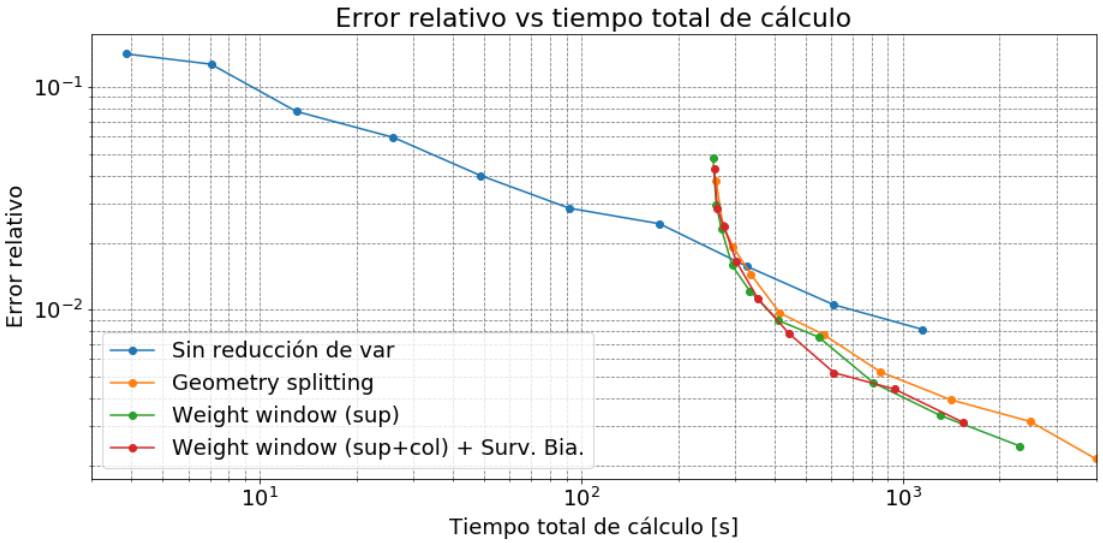

Figura 5.6: Error relativo en el pico de la Maxwelliana para distintas cantidades de partículas en función del tiempo total de cálculo de los mismos. Para el caso en el cual se aplicaron los métodos de reducción de varianza, el tiempo total de cálculo es la suma del tiempo de simulación de OpenMC, el tiempo empleado por el generador de mapa de importancias y el tiempo utilizado para la generación de las secciones eficaces que usa el generador. Por otro lado, para el caso sin reducción de varianza el tiempo total de cálculo es solamente el tiempo de simulación de OpenMC.

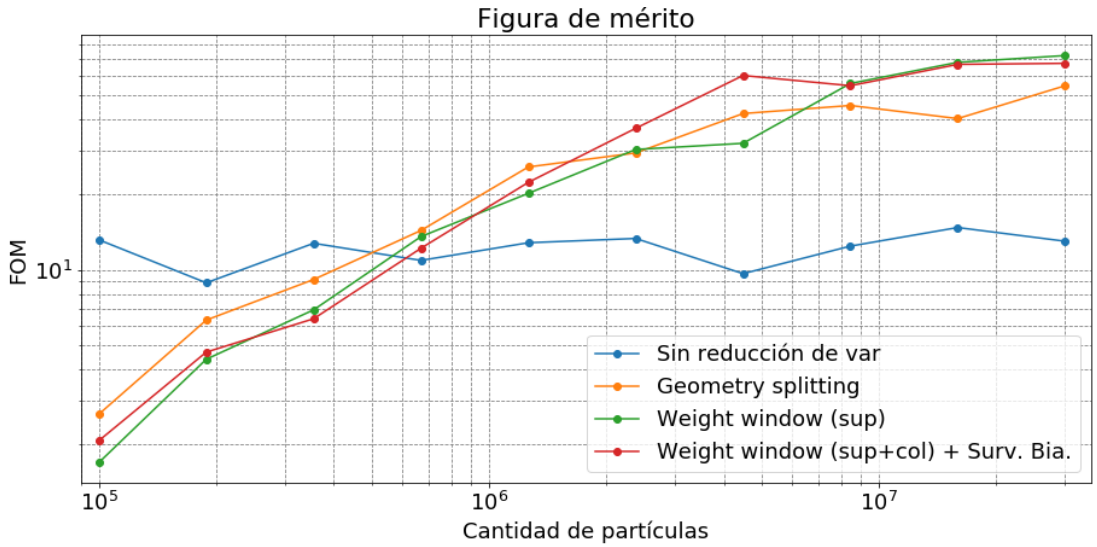

Figura 5.7: *FOM* tomando como error relativo en el pico de la Maxwelliana en función de la cantidad de partículas de fuente simuladas. El tiempo empleado para el cálculo de la FOM es el tiempo total de cálculo que se encuentra en la figura 5.6.

En la figura 5.6 se puede observar que para un tiempo de cálculo total mayor a aproximadamente 300 segundos, la utilización de los métodos de reducción de varianza permiten aumentar la figura de mérito. Esto es debido a que el tiempo empleado por el generador de mapa de importancias como así también el tiempo necesario para generar

las secciones eficaces a un grupo, tienden a ser despreciable con el aumento del tiempo de simulación producto de la disminución obtenida en la varianza.

Mediante el análisis llevado a cabo en este capítulo, se puede concluir que las técnicas de reducción de varianza implementadas como así también el generador de mapa de importancias permiten aumentar la figura de mérito.

También se comprobó que la técnica de reducción de varianza *Weight window* aplicado en las colisiones del neutrón junto con survival biasing funciona de forma adecuada.

En el caso de este modelo no se observan grandes diferencias entre las figuras de mérito obtenidas mediante la utilización de los tres métodos.

5.2. Benchmark numérico de fotones

Para estudiar el caso de transporte de fotones se modeló un problema benchmark planteado por Shultis en "An MCNP Primer" [25] como ejemplo para la comparación de los distintos métodos de reducción de varianza.

El mismo cuenta con una asignación de importancias para cada celda perteneciente a la geometría y por lo tanto se procederá en primer medida a utilizar dichas importancias para luego llevar a cabo un mapeo de las importancias con el generador de mapas de importancias implementado en el *API* de OpenMC.

5.2.1. Datos del benchmark

En la figura 5.8 se puede observar la geometría utilizada por el benchmark.

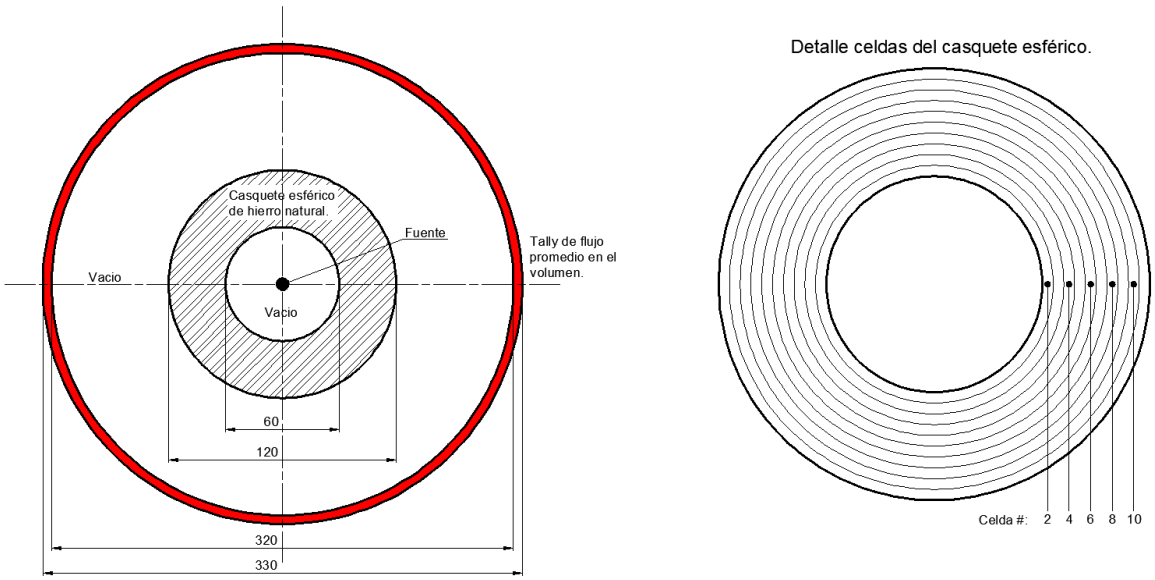

Figura 5.8: Geometría dada por el benchmark de fotones. Las dimensiones se encuentran en cm.

En el origen de coordenadas se cuenta con una fuente puntual de fotones, la cual es isotrópica y de 7 MeV. Dicha fuente se encuentra en vacío, rodeada de un casquete esférico de 30 cm de espesor. El material utilizado es *Fe* natural con una densidad de 7,86 $\frac{g}{cm^3}$ simulando un blindaje de fotones.

Además en la figura 5.8 se puede ver que se cuenta con un tally de flujo promedio en un volumen correspondiente a un casquete esférico que rodea al conjunto del blindaje y fuente. Con dicho tally se calculará la dosis utilizando los coeficientes de dosis ambientales equivalentes para fotones dado por el ICRP en 1987. En la siguiente figura se puede ver como varía el coeficiente de dosis en función de la energía:

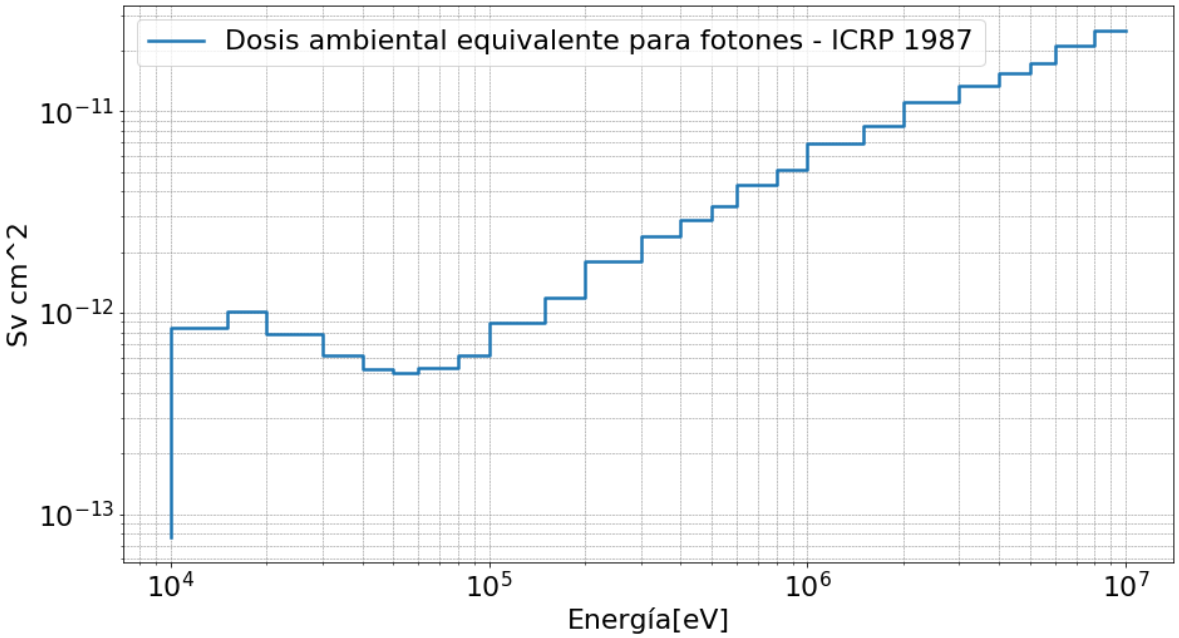

Figura 5.9: Coeficientes de dosis ambientales equivalentes para fotones. Se puede observar que los fotones de más alta energía son aquellos que más aportan a la dosis total.

Para la utilización de los métodos de reducción de varianza implementados se necesita que el mapa de importancias tenga una distribución espacial simulando una variación continua de la importancia de cada punto del espacio. Es por ello por lo cual a pesar de que el casquete esférico cuenta en todo su volumen con el mismo material, se subdivide al mismo en otros 10 casquetes esféricos concéntricos. En la tabla 5.2 se muestra el mapa de importancias utilizado, que coincide por el recomendado por Shultis en la especificación del benchmark.

Celda #	Importancia	Celda #	Importancia	Celda #	Importancia
1	1,0	6	16,0	10	256,0
2	1,0	7	32,0	11	512,0
3	2,0	8	64,0	12	512,0
4	4,0	9	128,0	13	512,0
5	8,0				

Tabla 5.2: Mapa de importancias utilizados para la simulación del problema.

La celda 1 es aquella que contiene a la fuente puntual, la celda 12 es la que se encuentra entre el casquete esférico y el volumen del tally, y la celda 13 el volumen del tally. Desde la celda 2 hasta la 11 se tienen todos los casquetes esféricos de *Fe* y se puede ver que la importancia de cada celda es el doble de la anterior, siendo esta una de las formas más comunes de asignación de importancias.

Por último, se cuenta con el input de MCNP (ver apéndice B.4) utilizado en el presente benchmark [25]. Se utilizó dicho input para obtener resultados utilizando MCNP para luego compararlo con los resultados dados por OpenMC. En ambos casos se utilizaron las biblioteca de secciones eficaces de fotones *eprdata14*.

5.2.2. Resultados

Para llevar a cabo un análisis de los métodos de reducción de varianza implementados en el código fuente se llevaron a cabo simulaciones utilizando los siguientes métodos:

- Sin método de reducción de varianza.
- *Geometry splitting*.
- *Weigh window* aplicado solamente cuando las partículas cruzan una superficie. Se lo denomina “Weight window (sup)”.

Los espectros del flujo de fotones y los errores relativos obtenidos con OpenMC y MCNP utilizando el mapa de importancias dado por la tabla 5.2 y 1×10^6 cantidad de partículas de fuente se encuentran en la figura 5.10 y 5.11 respectivamente.

Las simulaciones realizadas en OpenMC fueron llevadas a cabo utilizando los métodos de reducción de varianza *Geometry splitting* y *Weight window* en la superficie. Por otro lado, las simulaciones de MCNP fueron llevadas a cabo utilizando el método de reducción de varianza *Geometry splitting*. Para ambos casos se tienen las simulaciones sin la utilización de ninguna técnica para reducir la varianza.

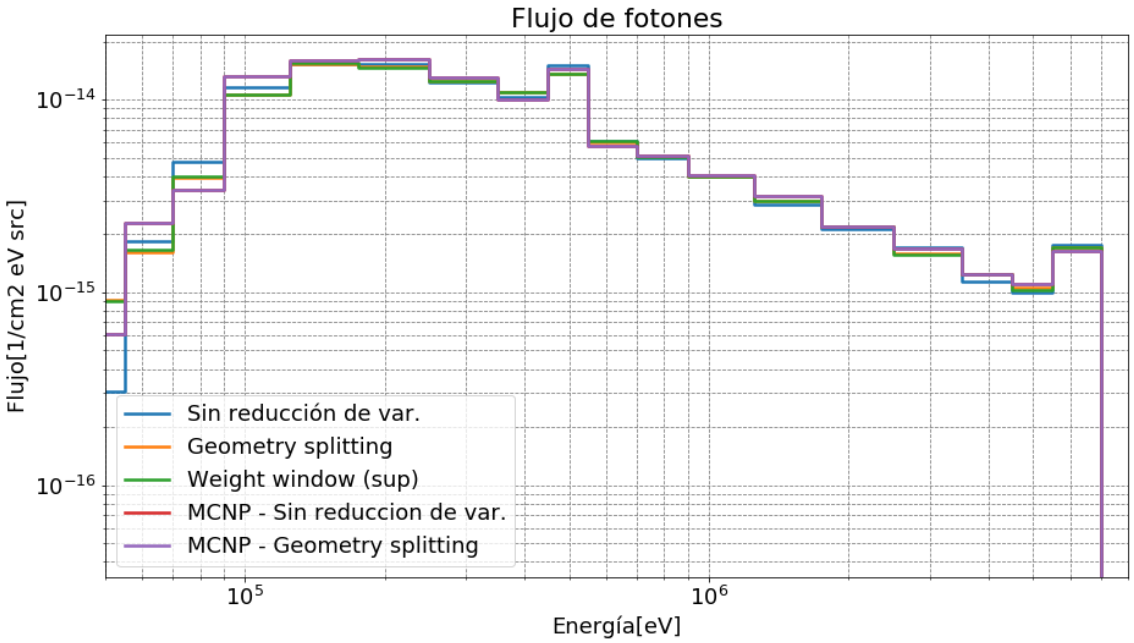

Figura 5.10: Espectro del flujo de fotones obtenido para las simulaciones con OpenMC y MCNP.

Se puede observar que los espectros del flujo de fotones son parecidos para energías mayores a 0,2 MeV lo cual verifica que el valor medio de la física del problema se conserva para cada método utilizado con OpenMC. Por otro lado, para energías menores a 0,2 MeV se ve que estas siguen la misma forma funcional pero analizando la figura 5.11, el error relativo para dichas energías es mayor que para el resto del espectro.

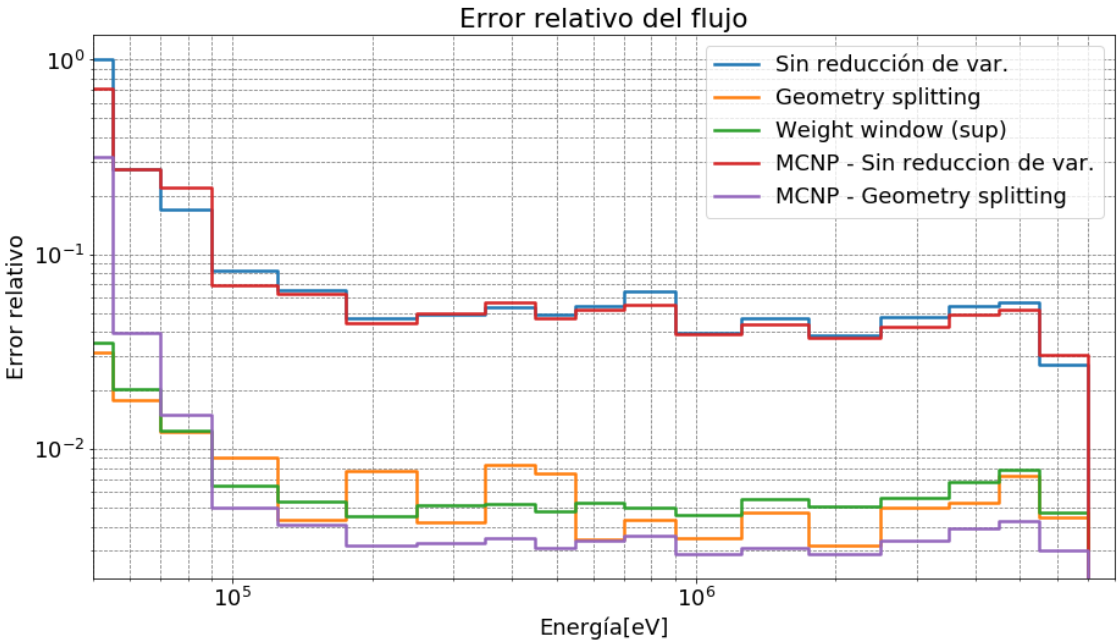

Figura 5.11: Error relativo del espectro del flujo de fotones obtenido para las simulaciones con OpenMC y MCNP.

Se verifica en esta figura que las técnicas de reducción de varianza implementadas efectivamente reducen la varianza del problema para una cantidad fija de partículas de fuente de fotones. Además se puede ver que el orden del error relativo del espectro obtenido es comparable con el resultante de la simulación realizada con MCNP. Por otro lado, para asegurar que ambos espectro son iguales, se grafica en la figura 5.12 los espectros de flujo de fotones obtenidos con sus bandas de error para ambos códigos de cálculo.

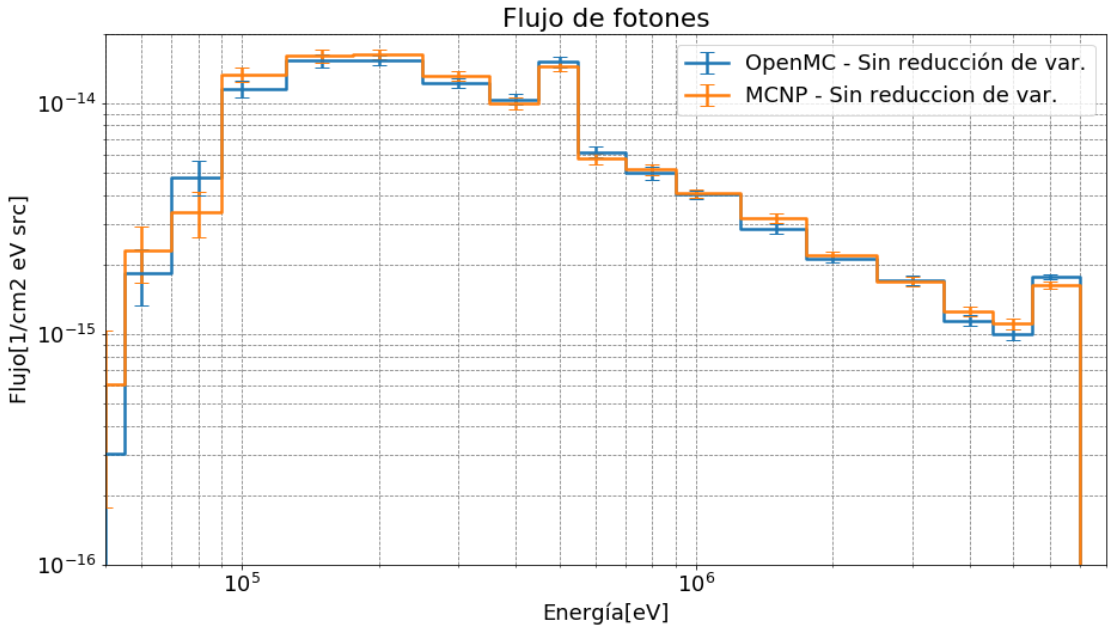

Figura 5.12: Espectro del flujo de fotones con los errores obtenidos utilizando ambos códigos de cálculo sin métodos para reducir la varianza.

No obstante, se busca reducir la varianza del problema de manera en la cual el tiempo de cálculo empleado para ello, no supere el tiempo de cálculo necesario para que se logre dicho error sin ninguna técnica implementada. Es por esta razón por la cual se realizaron simulaciones con distintas cantidades de partículas de fuente y utilizando los distintos métodos de reducción de varianza para cada caso. Para ello se tomaron los tiempos de cálculo empleados y el error relativo del espectro a una energía de 1MeV en cada caso. La elección de dicha energía para tomar el error relativo fue llevada a cabo de forma arbitraria, suponiendo que en dicha energía la varianza del error relativo es baja. En la siguiente figura se pueden observar los resultados obtenidos:

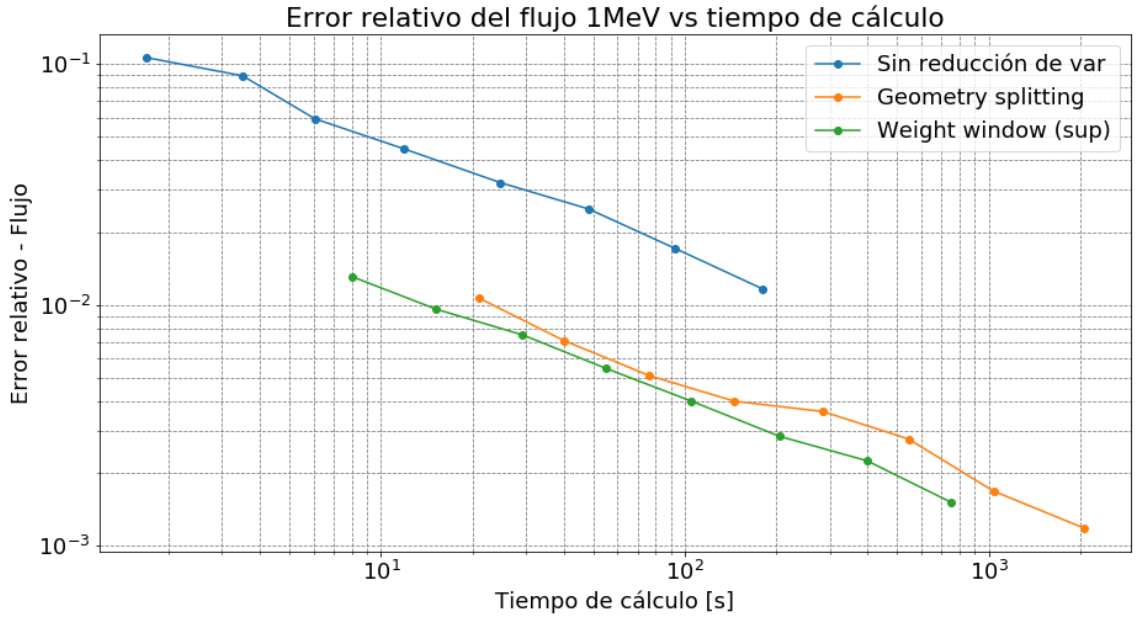

Figura 5.13: Error relativo del espectro del flujo de fotones a 1MeV en función del tiempo de cálculo empleado para distinta cantidad de partículas.

Se observa que para un mismo tiempo de cálculo, el error relativo obtenido es menor para el caso en el cual la simulación utiliza alguna de las dos técnicas de reducción de varianza respecto del caso en el que no. Esto conlleva finalmente a verificar el correcto funcionamiento de los métodos de reducción de varianza y la implementación llevada a cabo para el caso de fotones. Esto se traduce en un aumento en la figura de merito como se puede ver a continuación:

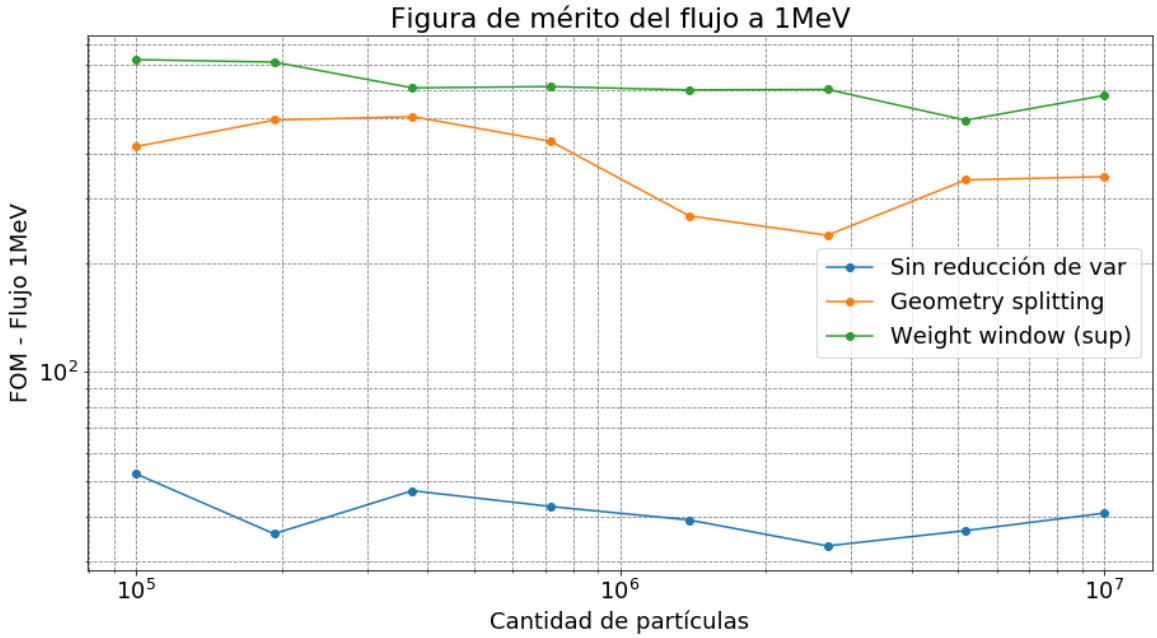

Figura 5.14: Figura de mérito en función de la cantidad de partículas simuladas. Se observa un aumento para ambos métodos de reducción de varianza aplicados.

Por último se calculó la dosis total equivalente utilizando los coeficientes dados por el ICRP para cada método de reducción de varianza y se los comparó con la dosis obtenida utilizando MCNP para los mismos casos.

Cantidad de partículas = 1e6			
Código	Método de reducción de varianza	Dosis total [Sv/gamma]	Error rel
OpenMC	Sin método	1,86e-19	0,05
	Geometry splitting	1,864e-19	0,003
	Weight window(sup)	1,860e-19	0,003
MCNP	Sin método	1,87e-19	0,02
	Geometry splitting	1,851e-19	0,002

Tabla 5.3: Dosis total obtenidas para ambos códigos de transporte mediante método Monte Carlo y los distintos métodos de reducción de varianza aplicados.

Se puede ver que el error relativo de la dosis obtenida es menor para el caso en el cual se aplican técnicas de reducción de varianza. También se verifica que esto se obtiene para un valor integral sobre el espectro del error relativo de la dosis a diferencia del análisis previo en el cual se tomaba el error de un punto del espectro.

Se tomaron los tiempos de cálculo y los errores relativos de la dosis total obtenida para distintas cantidades de partículas resultando en la figura 5.15. Luego se calculó la figura de mérito con estos datos obteniendo de esta manera la figura 5.16.

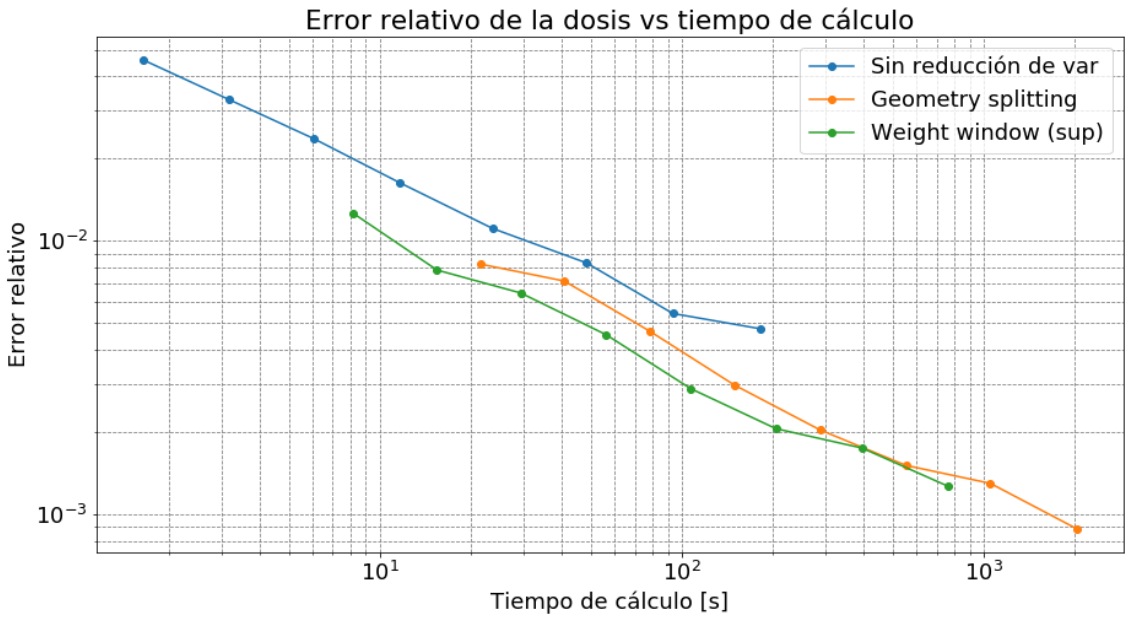

Figura 5.15: Error relativo de la dosis total en función del tiempo de cálculo empleado por OpenMC.

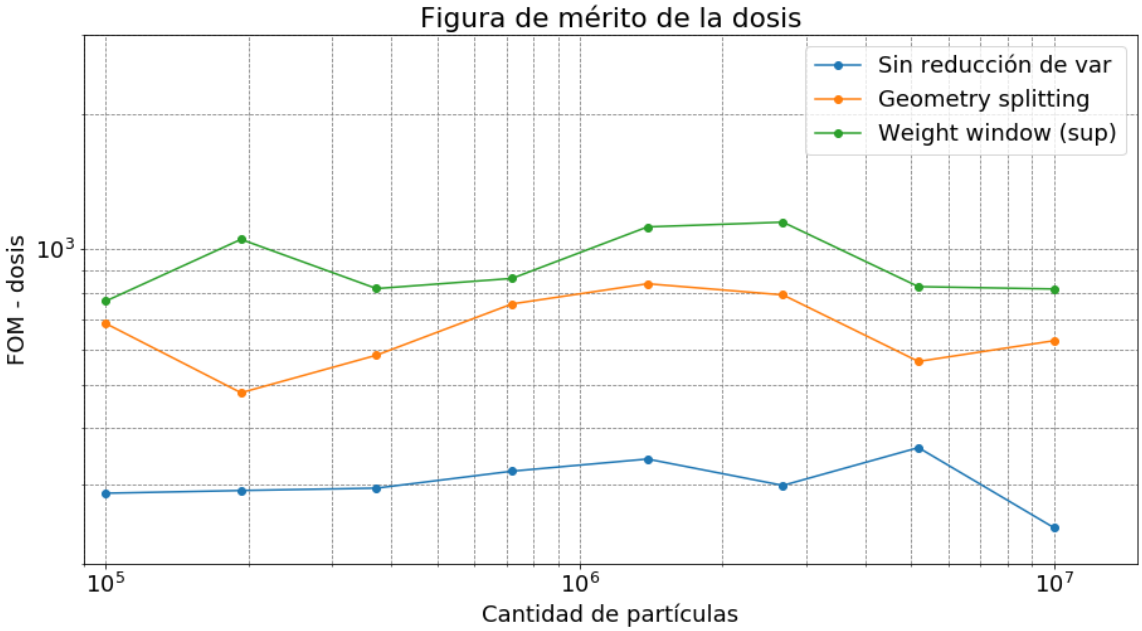

Figura 5.16: Figura de mérito de la dosis total en función de la cantidad de partículas simuladas. Se observa un aumento para ambos métodos de reducción de varianza aplicados.

Nuevamente, se observa una disminución del tiempo de cálculo necesario para obtener un error relativo en específico, resultando de esta manera en un aumento de la figura de merito para ambos casos.

5.2.3. Mapeo de importancias de la geometría

Se generó un nuevo mapa de importancias con el objetivo de verificar su funcionamiento en este caso en el cual las partículas son fotones. Para ello se utilizó el generador de mapa de importancias utilizando el modo del codigo denominado *total* el cual se corresponde a la utilización de la ecuación 4.1 para la estimación de la importancia. La sección eficaz total σ_t condensada fue determinada para un grupo de energías de 60keV a 7MeV ya que el espectro obtenido en la figura 5.10 se encuentra ubicado mayoritariamente dentro de dicho rango de energías. Buscando un error relativo del 1 % se determinó el siguiente mapa de importancias:

Celda #	Importancia	Celda #	Importancia	Celda #	Importancia
1	1,0	6	4,98	10	12,45
2	1,2	7	6,56	11	15,22
3	1,57	8	8,38	12	25,8
4	2,15	9	10,33	13	25,8
5	3,23				

Tabla 5.4: Mapa de importancias obtenidos mediante la utilización del generador de importancias.

Utilizando el mapa de importancias obtenido para la técnica *Geometry splitting*, los pesos correspondientes a los mismos para fijar las ventanas del método *Weight window* y la configuración recién descrita del benchmark se obtuvo el nuevo espectro de fotones y su error relativo en las figuras 5.17 y 5.18 respectivamente.

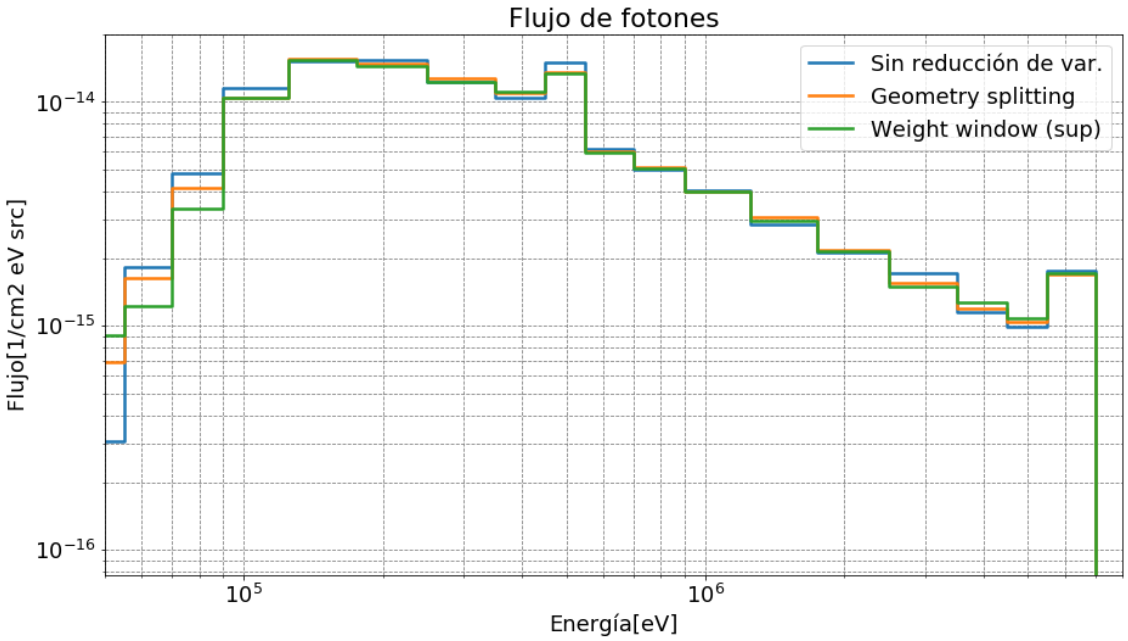

Figura 5.17: Espectros del flujo de fotones obtenido para las simulaciones con OpenMC utilizando el mapa de importancias generado.

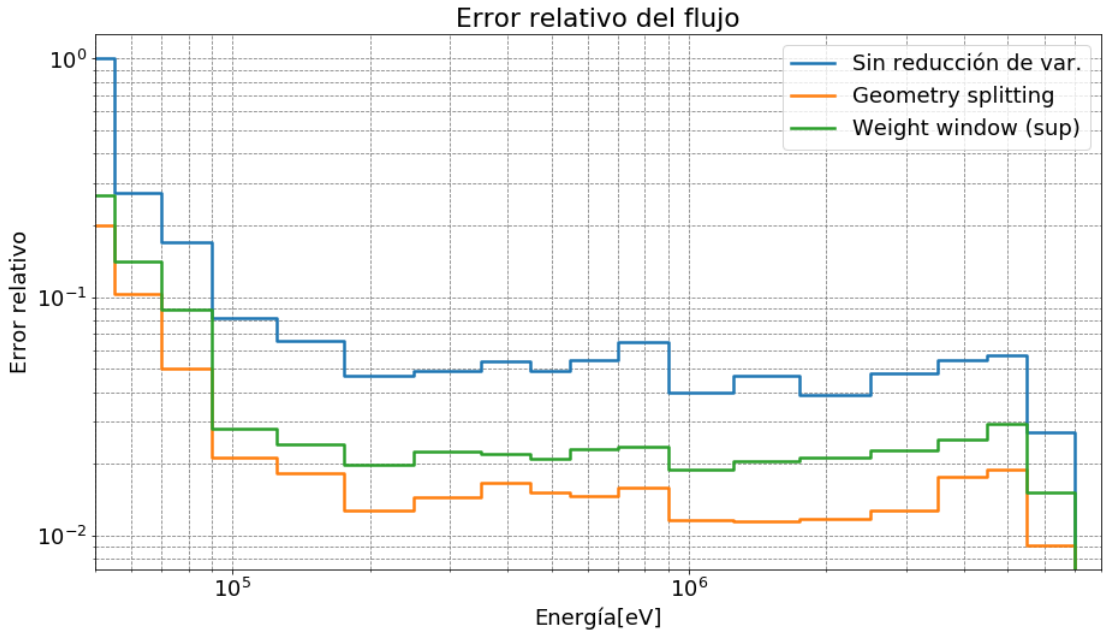

Figura 5.18: Error relativo de los espectros del flujo de fotones obtenido para las simulaciones con OpenMC utilizando el mapa de importancias generado.

Se puede observar que la forma funcional del flujo nuevamente es la misma en cada

caso y coinciden en la mayor parte del espectro. Por otro lado, vemos que el error relativo disminuye para ambos métodos de reducción de varianza, siendo estos menos de un órden de magnitud mayores al caso en el que el mapa de importancias utilizado fue dado por el benchmark. En la figura 5.19 se pueden observar los espectros de los errores relativos obtenidos para el caso en el cual se utiliza el generador de mapa de importancias y en el que no.

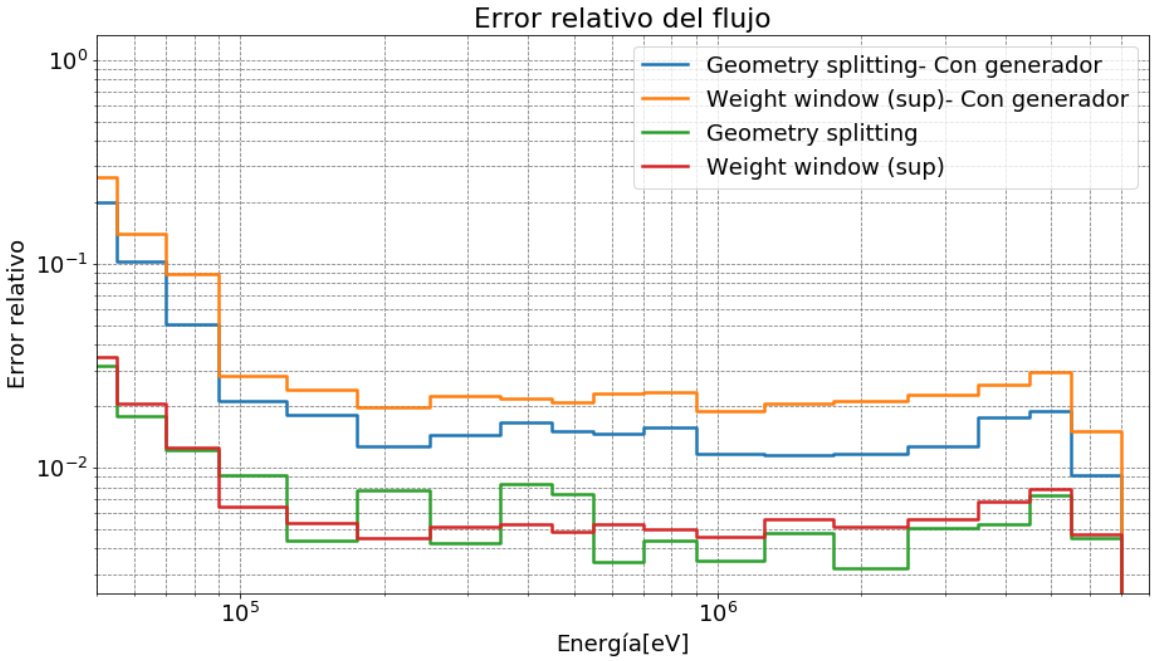

Figura 5.19: Comparación de los errores relativos obtenidos para el caso en el cual se utiliza el generador de mapa de importancias respecto de la asignación manual del mismo.

En la figura 5.20 se muestra la variación del error relativo del espectro a 1MeV en función del tiempo total de cálculo, mientras que en la figura 5.21 se graficó la variación de la FOM en función de la cantidad de partículas de fuente simuladas. El tiempo total de cálculo es la suma del tiempo empleado por el generador de mapa de importancias el cual fue de 74 segundos, el tiempo utilizado para calcular las secciones eficaces a un grupo el cual fue de 200 segundos y el tiempo de la simulación propiamente dicha.

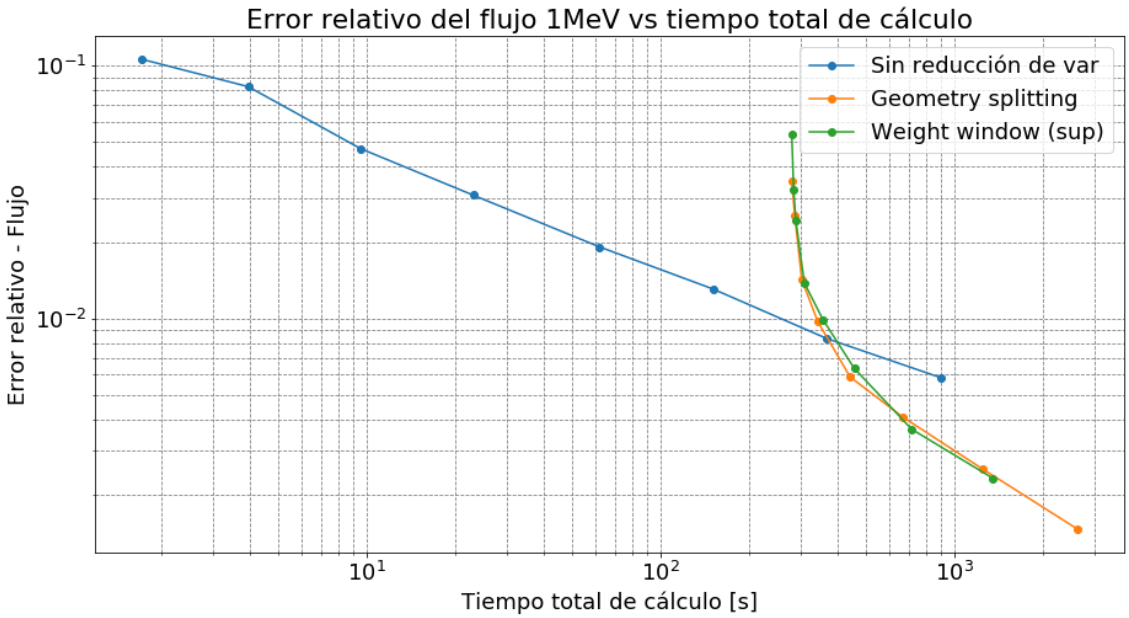

Figura 5.20: Error relativo del espectro del flujo de fotones a 1MeV en función del tiempo de cálculo empleado para distintas cantidades de partículas. El tiempo total de cálculo representa la suma del tiempo empleado por el generador de mapa de importancias, el tiempo utilizado para calcular las secciones eficaces y el tiempo de simulación de OpenMC, para el caso en el cual se aplicaron métodos de reducción de varianza. Por otro lado, para el caso sin reducción de varianza el tiempo total de cálculo es solamente el tiempo de simulación de OpenMC.

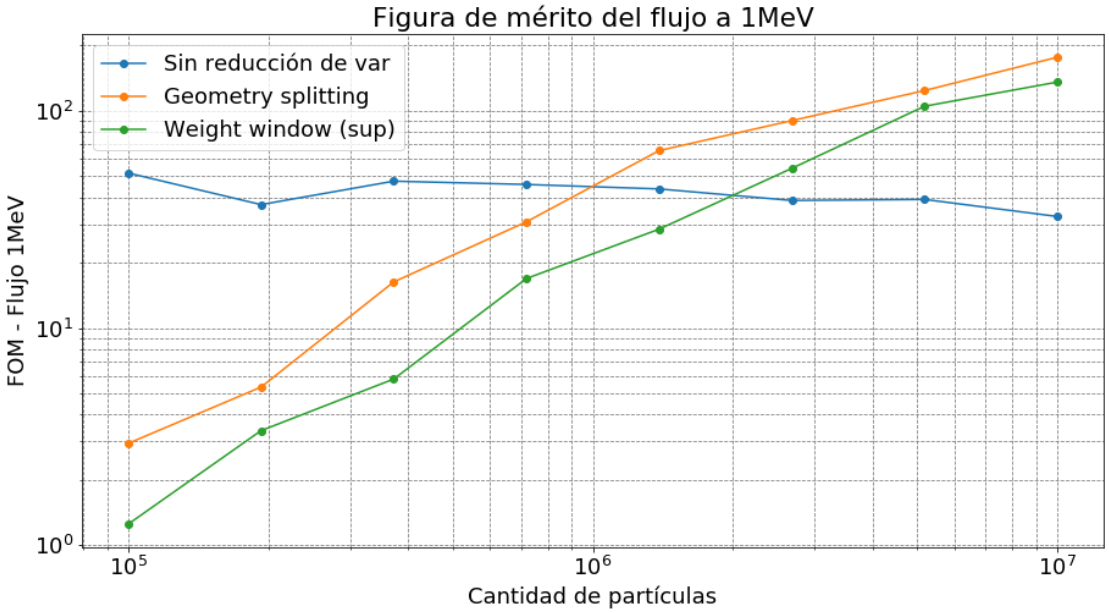

Figura 5.21: Figura de mérito en función de la cantidad de partículas simuladas. El tiempo empleado para el cálculo de la FOM es el tiempo total de cálculo que se encuentra en la figura 5.20.

En la figura 5.20 se puede observar que para un tiempo total de cálculo mayor a 400 segundos aproximadamente la figura de mérito es mayor para los casos en los cuales se utilizan técnicas de reducción de varianza. Esto se debe a que el tiempo empleado

por el generador es constante, mientras que el tiempo de simulación aumenta con la cantidad de partículas simuladas haciendo que la figura de mérito aumente en estos casos.

Por último, se calculó la dosis total y se obtuvieron los siguientes resultados:

Cantidad de partículas = 1e6			
Código	Método de reducción de varianza	Dosis total [Sv/gamma]	Error rel
OpenMC	Sin método	1,86e-19	0,05
	Geometry splitting	1,85e-19	0,01
	Weight window(sup)	1,86e-19	0,01

Tabla 5.5: Dosis total y el error relativo de la misma obtenidos utilizando OpenMC y los distintos métodos de reducción de varianza.

Se puede ver que el error relativo es menor en ambos casos respecto de la simulación en la cual no se utilizó ninguna técnica implementada. Analizando el tiempo de cálculo necesario para cada simulación y error relativo de la dosis en función de la cantidad de partículas se obtuvieron los siguiente gráficos:

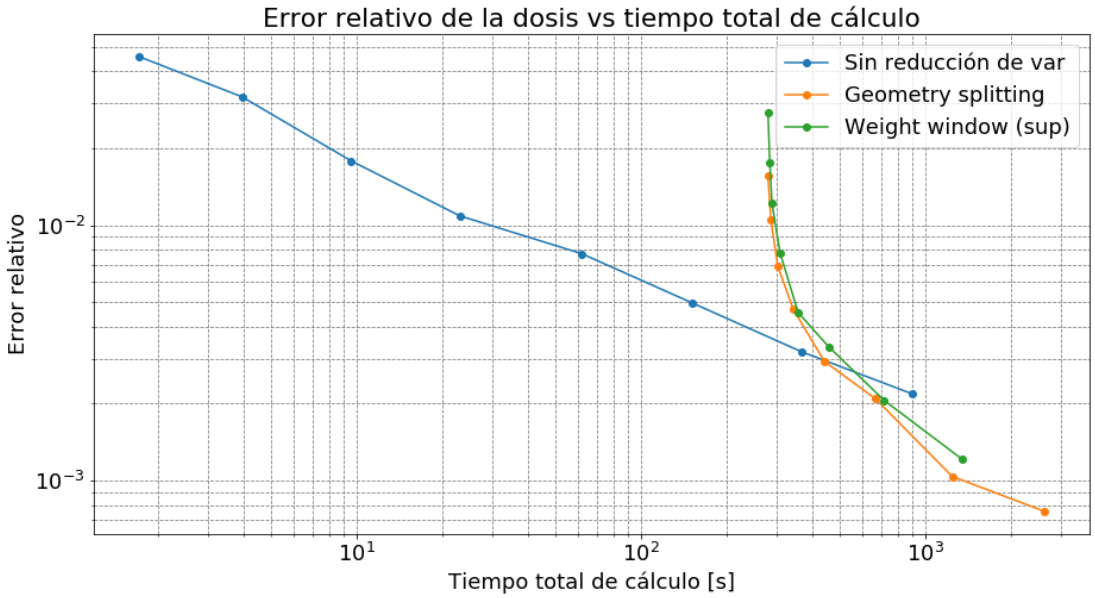

Figura 5.22: Error relativo de la dosis total en función del tiempo total de cálculo.

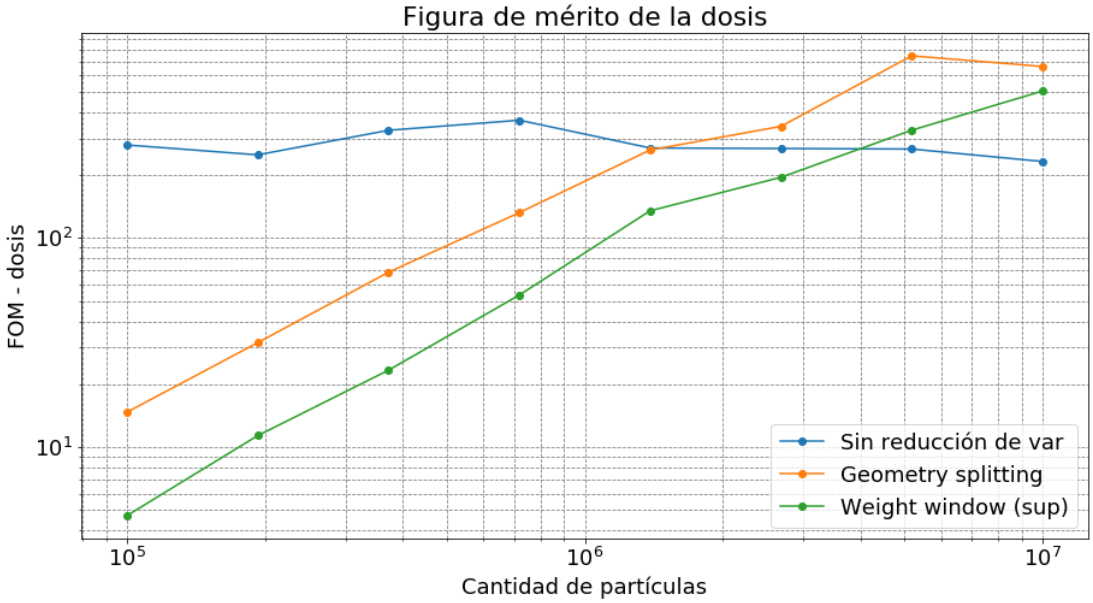

Figura 5.23: Figura de mérito en función de la cantidad de partículas de fuente simuladas.

Se observa que el tiempo total de cálculo para el cual, llevar a cabo la estimación de importancias con OpenMC aplicando las técnicas de reducción de varianza, reduce la FOM para tiempos mayores a aproximadamente 500 segundos.

De esta forma se comprueba el correcto funcionamiento de las técnicas de reducción de varianza implementadas para el caso en el cual la partícula transportada son fotones.

Se verifica que la figura de merito aumenta independientemente de la forma en la cual se toma el error relativo, es decir de forma puntual (una energía dada) o de forma integral (error de la dosis total).

No se observan grandes diferencias entre las figuras de merito obtenidas mediante ambas técnicas. Para llevar a cabo este análisis se deberán analizar geometrías más complicadas que la presentada en este benchmark.

Capítulo 6

Conclusiones

Se estudió la utilización del código OpenMC para el cálculo de blindajes. Para esto se validó el código y se implementaron métodos de reducción de varianza.

La validación del código para cálculos de criticidad (ver sección 2.1) se llevó a cabo utilizando el benchmark IEU-COMP-FAST-001 el cual cuenta con resultados correspondientes a simulaciones con MCNP, como así también experimentales. Utilizando OpenMC se obtuvo el mismo k_{eff} con una incerteza de cálculo de 10 pcm que el resultado de MCNP y en el caso del resultado experimental se obtuvo una diferencia porcentual en el k_{eff} igual a 0,08 %.

La validación de transporte de neutrones se hizo con el benchmark de la esfera de hierro Oktavian de la compilación SINBAD (ver sección 2.2) el cual presenta resultados experimentales. También se verificó la equivalencia del espectro obtenido con OpenMC, analizando el mismo problema utilizando MCNP.

Se verificó la física del transporte de fotones de OpenMC (ver sección 2.3) utilizando el benchmark numérico planteado en el informe técnico ANL/MCS-TM-381 del *Argonne National Laboratory*. En el mismo se comprobó que la diferencia porcentual entre el espectro de fotones obtenido con OpenMC y MCNP se encuentran dentro de una banda del 4 % de diferencia porcentual lo que es consistente con los resultados presentados en dicho informe.

Se implementaron las técnicas de reducción de varianza Geometry splitting y Weight window en el código fuente, como así también en el API de OpenMC. Se desarrolló un generador de mapa de importancias en el API de OpenMC el cual permite calcular en forma automática los parámetros necesarios para la aplicación de estas técnicas de reducción de varianza.

Los métodos de reducción de varianza implementados en OpenMC fueron validados analizando un problema ejemplo para neutrones y un benchmark numérico de fotones (ver capítulo 5), en los cuales además se utilizó el generador de mapa de importancias. Se comprobó que el valor medio del espectro en cada caso permanece constante al

utilizar las técnicas de reducción de varianza. Por otro lado, si bien la utilización del generador de mapa de importancias aumenta el tiempo total de cálculo, se comprobó para cada caso que para un tiempo de simulación dado esto se compensa con el aumento de la precisión en el cálculo llevando a un incremento de la figura de mérito respecto de la simulación sin métodos de reducción de varianza. De esta forma se verificó el funcionamiento de las técnicas implementadas para el caso de transporte de neutrones y de fotones.

Como trabajo a futuro se plantea la implementación de otras técnicas de reducción de varianza, que no pudieron ser implementadas durante el proyecto integrador por falta de tiempo. Dentro de estas se encuentran los métodos de source biasing, transformaciones exponenciales y multiplicación de fuentes superficiales, entre otros. También se plantea la optimización de los programas implementados para reducir el tiempo de cálculo y aumentar la figura de mérito.

Apéndice A

Demostraciones

A.1. Prueba de la varianza de una combinación lineal

$$
\begin{aligned}
\operatorname{var}\left(\sum_{i=1}^{n} a_i X_i\right) &= E\left[\left(\sum_{i=1}^{n} a_i X_i\right)^2\right] - \left(E\left[\sum_{i=1}^{n} a_i X_i\right]\right)^2 && \text{Definicion de varianza} \\
&= E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j X_i X_j\right] - \left(E\left[\sum_{i=1}^{n} a_i X_i\right]\right)^2 && \text{Propiedades basicas de la suma} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j E[X_i X_j] - \left(\sum_{i=1}^{n} a_i E[X_i]\right)^2 && \text{Linealidad de la esperanza} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j E[X_i X_j] - \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j E[X_i]E[X_j] && \text{Propiedades de la sumatoria} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \left(E[X_i X_j] - E[X_i]E[X_j]\right) && \text{Combinacion de las sumas} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \operatorname{cov}(X_i, X_j) && \text{Por definicion de covarianza} \\
&= \sum_{i=1}^{n} a_i^2 \operatorname{var}(X_i) + 2\sum_{i=1}^{n}\sum_{j\,:\,j>i}^{n} a_i a_j \operatorname{cov}(X_i, X_j) && \text{Re ordenando la sumatoria}
\end{aligned}
$$

Apéndice B

Códigos utilizados

B.1. Códigos implementados para geometry splitting

A continuación, se puede observar la implementación llevada a cabo en el archivo `particle.cpp` del código fuente para la funcion *void Particle::get_ importance();*:

```
void
Particle::get_importance(){

    if (this->imp_ == -1.0){
        Position r {this->r()};
        this->r() += TINY_BIT * this->u();
        const Cell& c {*model::cells[this->coord_[this->n_coord_-1].cell]};
        this->r() = r;

        if (c.importance_.size() > 1) {
          this->imp_ = c.importance_[this->cell_instance_];
        } else {
          this->imp_ = c.importance_[0];
        }
    }

    else{

                // Asigno la importancia como la de la celda anterior.
        this->imp_last_ = this->imp_;

        Position r {this->r()};
        this->r() += TINY_BIT * this->u();
        const Cell& c {*model::cells[this->coord_[this->n_coord_-1].cell]};
        this->r() = r;


        if (c.importance_.size() > 1) {
          this->imp_ = c.importance_[this->cell_instance_];
        } else {
```

```
          this->imp_ = c.importance_[0];
        }

    }

}
```

A continuación, se puede observar la implementación llevada a cabo en el archivo `particle.cpp` del código fuente para la función *void Particle::geometry_splitting();*:

```
void
Particle::geometry_splitting(){

    if (this->imp_> this->imp_last_) {
      int32_t n;
      int32_t i;
      double_t prob;

      n = std::floor(this->imp_/this->imp_last_);
      prob = this->imp_/this->imp_last_ - n;

      if (prn() < prob ) n++;

      for (i=0; i<n-1;i++){
          simulation::secondary_bank.emplace_back();
          auto& bank {simulation::secondary_bank.back()};
          bank.particle = this->type_;
          bank.wgt = this->wgt_*this->imp_last_/this->imp_;
          bank.r = this->r();
          bank.u = this->u();
          bank.E = this->E_;
          this->n_bank_second_ += 1;
      }
      this->wgt_ = this->wgt_*this->imp_last_/this->imp_;

    } else{
      if (prn() < this->imp_ / this->imp_last_) {
        this->wgt_last_ = this->wgt_;
        this->wgt_ = this->wgt_*this->imp_last_/this->imp_;
      } else {
        //std::cout<<"It's dead, Jim\n";
        this->imp_ = -1;
        this->imp_last_ = -1;
        this->wgt_ = 0.;
        this->wgt_last_ = 0.;
        this->alive_ = false;
      }
    }
}
```

B.2. Código implementado de la ruleta rusa para control poblacional

A continuación, se puede observar la implementación llevada a cabo en el archivo `physics_common.cpp` del código fuente:

```
//Ruleta rusa para control poblacional.
void russian_roulette(Particle* p)
{
  if (p->wgt_ < settings::weight_cutoff / p->imp_ ) {
    if (prn() < p->wgt_ * p->imp_ / settings::weight_survive) {
      p->wgt_last_ = p->wgt_;
      p->wgt_ = settings::weight_survive / p->imp_ ;
    } else {
      p->wgt_ = 0.;
      p->wgt_last_ = 0.;
      p->lower_weight_ = -1;
      p->upper_weight_ = -1;
      p->survival_weight_ = -1;
      p->imp_ = -1;
      p->imp_last_ = -1;
      p->alive_ = false;
    }
  }
}
```

B.3. Códigos implementados para weight window

A continuación, se puede observar la implementación llevada a cabo en el archivo `particle.cpp` del código fuente para la función *void Particle::get_window();*:

```
void
Particle::get_window(){
    Position r {this->r()};
    this->r() += TINY_BIT * this->u();
    const Cell& c {*model::cells[this->coord_[this->n_coord_-1].cell]};
    this->r() = r;

    if (c.upper_weight_.size() > 1) {
      this->upper_weight_ = c.upper_weight_[this->cell_instance_];
      this->lower_weight_ = c.lower_weight_[this->cell_instance_];
      this->survival_weight_ = c.survival_weight_[this->cell_instance_];
    } else {
      this->upper_weight_ = c.upper_weight_[0];
      this->lower_weight_ = c.lower_weight_[0];
      this->survival_weight_ = c.survival_weight_[0];
    }
}
```

A continuación, se puede observar la implementación llevada a cabo en el archivo `particle.cpp` del código fuente para la funcion *void Particle::weight_window();*:

```
void
Particle::weight_window(){

    int32_t n;
    int32_t i;
    double_t prob;

    if(this->wgt_ > this->upper_weight_){
        n = std::floor(this->wgt_/this->survival_weight_);
        prob = (this->wgt_/this->survival_weight_) - n;

        if (prn() < prob ) n++;

        for (i=0; i<n-1;i++){
            simulation::secondary_bank.emplace_back();
            auto& bank {simulation::secondary_bank.back()};
            bank.particle = this->type_;
            bank.wgt = this->survival_weight_;
            bank.r = this->r();
            bank.u = this->u();
            bank.E = this->E_;
            this->n_bank_second_ += 1;
        }
      this->wgt_ = this->survival_weight_;
    }
    if(this->wgt_ < this->lower_weight_){
        russian_roulette_weight_window(this);
    }

}
```

B.4. Input de MCNP para el benchmark de fotones

```
Point isotropic 7-MeV photon sources in iron shell: (analog base case):
c ******************** BLOCK 1: CELL CARDS ****************************
c GEOMETRY:     X isotropic point source (7-MeV)
c               D ambient dose 100 cm from outer shield surface (160 cm)
c                 iron shield 30-cm thick (r=30 to 60 cm)
c              (without shield, dose is 6.013x10^{-17} Sv/gamma)
c
c                   z-axis ^
c                          |     \       \         void
c                          |       \ Fe    \
c                          | void   |       |
c                          X -------|-------|--------D----> x-axis
c                      source       |       |
```

```
c                                    /       /
c                                   /       /
c
c ********************* BLOCK 1: CELLS *********************************
  10 0             -10         imp:p=1     $ inside of shield
  20 1    -7.86    10 -11      imp:p=1     $ iron shell
  21 1    -7.86    11 -12      imp:p=2     $ iron shell
  22 1    -7.86    12 -13      imp:p=4     $ iron shell
  23 1    -7.86    13 -14      imp:p=8     $ iron shell
  24 1    -7.86    14 -15      imp:p=16    $ iron shell
  25 1    -7.86    15 -16      imp:p=32    $ iron shell
  26 1    -7.86    16 -17      imp:p=64    $ iron shell
  27 1    -7.86    17 -18      imp:p=128   $ iron shell
  28 1    -7.86    18 -19      imp:p=256   $ iron shell
  29 1    -7.86    19 -20      imp:p=512   $ iron shell
  30 0             20 -50      imp:p=512   $ void outside shld and inside detect
  40 0             50 -51      imp:p=512 vol=1.659285E06  $ detector
  41 0             51 -100     imp:p=512   $ void past detector
  50 0              100        imp:p=0     $ vacuum outside problem boundary

c *********************     BLOCK 2: SURFACE CARDS ************************
  10 so 30.0                 $ inner shield surface
  11 so 33.0                 $ inner shield surface
  12 so 36.0                 $ inner shield surface
  13 so 39.0                 $ inner shield surface
  14 so 42.0                 $ inner shield surface
  15 so 45.0                 $ inner shield surface
  16 so 48.0                 $ inner shield surface
  17 so 51.0                 $ inner shield surface
  18 so 54.0                 $ inner shield surface
  19 so 57.0                 $ inner shield surface
  20 so 60.0                 $ outer shield surface
  50 so 160.0                $ detector surface
  51 so 165.0                $ detector surface
 100 so 10.E+02              $ spherical problem boundary (at 10 m)

c ********************* BLOCK 3: DATA CARDS ****************************
SDEF erg=7.00 par=2          $ 7-Mev pt photon source at origin
c
mode p
phys:p 100 1 1      $ no bremsstrahlung; no coherent scattering
nps 100000          $ 100000 particle cutoff
f4:p    40          $ tally on cell 40 as ambient dose
c
c ---- Photon ambient dose equivalent H*(10mm) Sv cm^2; ICRP [1987]
de4     0.100E-01 0.150E-01 0.200E-01 0.300E-01 0.400E-01 0.500E-01 &
        0.600E-01 0.800E-01 0.100E+00 0.150E+00 0.200E+00 0.300E+00 &
        0.400E+00 0.500E+00 0.600E+00 0.800E+00 0.100E+01 0.150E+01 &
        0.200E+01 0.300E+01 0.400E+01 0.500E+01 0.600E+01 0.800E+01 &
        0.100E+02
df4     0.769E-13 0.846E-12 0.101E-11 0.785E-12 0.614E-12 0.526E-12 &
```

```
        0.504E-12 0.532E-12 0.611E-12 0.890E-12 0.118E-11 0.181E-11 &
        0.238E-11 0.289E-11 0.338E-11 0.429E-11 0.511E-11 0.692E-11 &
        0.848E-11 0.111E-10 0.133E-10 0.154E-10 0.174E-10 0.212E-10 &
        0.252E-10
c
f14:p    40          $ tally on cell 40 as flux
c
c ----
e14     0.100E-01 0.150E-01 0.200E-01 0.300E-01 0.400E-01 0.500E-01 &
        0.600E-01 0.800E-01 0.100E+00 0.150E+00 0.200E+00 0.300E+00 &
        0.400E+00 0.500E+00 0.600E+00 0.800E+00 0.100E+01 0.150E+01 &
        0.200E+01 0.300E+01 0.400E+01 0.500E+01 0.600E+01 0.800E+01 &
        0.100E+02
c
c --- Natural iron (density 7.86 g/cm^3)
m1     26000 -1.00000
```

Bibliografía

[1] Smith, M. A. Zpr-6 assembly 6a: A cylindrical assembly with uranium oxide fuel and sodium with a thick depleted-uranium blanket. En: International Handbook of Evaluated Criticality Safety Benchmark Experiments (ICSBEP), IEU-COMP-FAST-001. Organization for Economic Co-operation and Development-Nuclear Energy Agency (OECD-NEA), 2016.

[2] Hashikura, H. Measurements of neutron leakage spectra from 50.32 cm radius iron sphere. En: SINBAD Shielding Benchmark Experiments Status and Planned Activities. Organization for Economic Co-operation and Development-Nuclear Energy Agency (OECD-NEA).

[3] Team, X.-. M. C. La-ur-03-1987 - mcnp — a general monte carlo n-particle transport code, version 5 volume i: Overview and theory. Inf. téc., Los Alamos National Lab., 24 de Abril, 2003.

[4] Romano, P. K., Horelik, N. E., Herman, B. R., Nelson, A. G., Forget, B., Smith, K. Openmc: A state-of-the-art monte carlo code for research and development. En: SNA+ MC 2013-Joint International Conference on Supercomputing in Nuclear Applications+ Monte Carlo, pág. 06016. EDP Sciences, 2014.

[5] GNU. Software libre. `https://www.gnu.org/philosophy/free-sw.es.html`.

[6] OpenMC. Biblioteca de secciones eficaces microscopicas. `https://openmc.org/official-data-libraries/`.

[7] Mauricio Ezequiel Debárbora. OpenMC con métodos de reducción de varianza implementados. `https://github.com/MauriDeb/Desarrollo_OpenMC/tree/Desarrollo`.

[8] OpenMC. Manual OpenMC: Materiales. `https://docs.openmc.org/en/stable/usersguide/materials.html`.

[9] OpenMC. Manual OpenMC: Geometria. `https://docs.openmc.org/en/stable/usersguide/geometry.html`.

[10] OpenMC. Manual OpenMC: Settings. `https://docs.openmc.org/en/stable/usersguide/settings.html`.

[11] OpenMC. Manual OpenMC: Tallies. `https://docs.openmc.org/en/stable/methods/tallies.html`.

[12] Lewis, E. E., Miller, W. F. Computational methods of neutron transport, 1984.

[13] Hoogenboom, J. E., Légrády, D. A critical review of the weight window generator in mcnp. En: Proc. Monte Carlo 2005 Topical Meeting: The Monte Carlo Method–Versatility Unbounded in a Dynamic Computing World. 2005.

[14] Sobol. The monte carlo method, 1974.

[15] OpenMC. Manual OpenMC: Tallies. `https://docs.openmc.org/en/stable/methods/tallies.html`.

[16] Bell, G. I., Glasstone, S. Nuclear reactor theory. Inf. téc., US Atomic Energy Commission, Washington, DC (United States), 1970.

[17] Oberkampf, W. L., Trucano, T. G. Verification and validation benchmarks. *Nuclear engineering and Design*, **238** (3), 716–743, 2008.

[18] Milocco, A. The quality assessment of the oktavian benchmark experiments. *IJS-DP-10214, Institut Jošef Stefan, Reactor Physics Department, Ljubljana*, 2009.

[19] Lund, A. L., Romano, P. K. Implementation and validation of photon transport in openmc. Inf. téc., Argonne National Lab.(ANL), Argonne, IL (United States), 2018.

[20] OpenMC. Manual OpenMC: Física de neutrones. `https://docs.openmc.org/en/stable/methods/neutron_physics.html`.

[21] Lux, I. Monte Carlo particle transport methods. CRC press, 2018.

[22] Nowak, M. Accelerating Monte Carlo particle transport with adaptively generated importance maps. Tesis Doctoral, Université Paris-Saclay (ComUE), 2018.

[23] Petit, O., Hugot, F.-X., Lee, Y.-K., Jouanne, C., Mazzolo, A. Tripoli-4 version 4 user guide. Inf. téc., CEA Saclay, 2008.

[24] Lamarsh, J. R. Introduction to nuclear reactor theory. Addison-Wesley, 1966.

[25] Shultis, J. K., Faw, R. E. An mcnp primer. Inf. téc., 2011.

Agradecimientos

A mis ejemplos a seguir en la vida, mi Mamá y mi Papá, las personas que siempre me alentaron y estuvieron para apoyarme cuando los necesité. A Matías, mi hermano, a quien admiro mucho y que desde el primer momento me apoyó en este camino.

A Nacho Marquez le agradezco el esfuerzo puesto para guiarme a lo largo del proyecto y por estar siempre dispuesto a ayudarme cuando lo necesité. A Ariel Marquez por toda la ayuda que me brindó a lo largo de este trabajo.

A Brian Escalante, Sebastián Calvera, Agustín Bernardo, Nahuel Nuñez, Marianella Montaño, Nicolás Morgan y Sebastián Sigvard, conocerlos fue una de las partes mas lindas de estos últimos tres años, sin su apoyo y compañía este camino hubiese sido imposible.

A Rodrigo Manzano, mi amigo incondicional con el que siempre pude contar a pesar de la distancia. A Norberto Schmidt por su amistad y por estar siempre que lo necesité.